НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

До захисту допущено
Завідувач кафедри
_____ *О.В. Коваль*
(підпис) (ініціали, прізвище)
“___”_____ 2020 р.

Дипломна робота
на здобуття ступеня бакалавра

з напряму підготовки: ***122 “Комп’ютерні науки”***
на тему:
«СТВОРЕННЯ ПЛАТФОРМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА»

Виконав: студент 4 курсу, групи ТР-62
Коцубанов Гліб Костянтинович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: *Кузьміних В.О., к.т.н., доцент* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент: *Сенченко В.Р., к.т.н.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Кафедра *автоматизації проектування енергетичних процесів і систем*
Рівень вищої освіти *перший , бакалаврський*
Напрям підготовки *122“ Комп’ютерні науки ”*
за спеціалізацією — *Комп’ютерне геометричне моделювання процесів та систем*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020 р.

ЗАВДАННЯ
на дипломну роботу

Коцубанова Гліба Костянтиновича
(прізвище, ім’я, по батькові)

1. Тема роботи «***СТВОРЕННЯ ПЛАТФОРМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА***» керівник роботи *Кузьміних В.О., к.т.н., доцент*
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)
затверджена наказом вищого навчального закладу від 25.05.2020р. № 1268-с
2. Строк подання роботи: __15.06.2020__
3. Вихідні дані до роботи Мова програмування Python, платформа написання програмного коду PyCharm
4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Проаналізувати існуючі інструментальні засоби та бази даних, які дозволяють отримати необхідну інформацію, запропонувати ефективний підхід для вирішення задачі побудови платформи для пошуку, вилучення та збереження інформації для оцінки міжнародної науково-технічної діяльності, розробити алгоритми та програмні засоби для надання аналітику необхідної інформації, впорядкованої за релевантністю відповідно до запиту.
5. Перелік ілюстративного матеріалу "Актуальність", "Мета", "Пов’язана проблематика", "Схема платформи та додатку", "Функції платформи", " Функції платформи реалізовані наступним шляхом", "Схема збереження даних у pandas", "Функції додатку", "Алгоритм додатку", "Інтерфейс користувача", "Використання синтаксису «еластичного пошуку»", "Використані програмні засоби" ,"Висновки".

11 жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	10.10.2019-20.12.2019	
2	Розробка архітектури та загальної структури системи	21.12.19-20.01.2020	
3.	Розробка структур окремих підсистем	21.01.2020-20.02.2020	
4.	Програмна реалізація системи	21.02.2020-17.05.2020	
5.	Оформлення пояснювальної записки	18.05.2020-20.05.2020	
6.	Захист програмного продукту	25.05.2020	
7.	Передзахист	26.05.2020	
8.	Захист	20-21.06.2020	

Студент: *Коцубанов Г.К.* _____
(прізвище та ініціали) (підпис)

Керівник роботи: *Кузьміних В.О.* _____
(прізвище та ініціали) (підпис)

АНОТАЦІЯ

У виконаній дипломній роботі проаналізовано існуючі інструментальні засоби та бази даних, які дозволяють отримати необхідну інформацію. На базі отриманої інформації запропоновано ефективний підхід для вирішення задачі побудови платформи для пошуку, вилучення та збереження інформації для оцінки міжнародної науково-технічної діяльності.

Метою є розробка платформи пошуку, вилучення та збереження інформації за вузьконаправленим запитом, та подальша її обробка з метою впорядкування у релевантну послідовність.

Робота складається зі вступу, шести розділів та висновків. У вступі розкрито тему пошуку та використання інформації в межах міжнародного науково-технічного співробітництва. В першому розділі представлено постановку задачі платформи для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва. В другому розділі представлено існуючі варіанти рішень, подібні до платформи для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва. В третьому розділі запропоновано способи виконання поставленої задачі. В четвертому розділі описано засоби розробки програмного продукту. В п’ятому розділі описано розроблений програмний продукт. В шостому розділі наведено сценарій використання програмного продукту.

Ключові слова: платформа, релевантність, критерії, запит.

ABSTRACT

In the completed thesis was analyzed the existing tools and databases that allow to obtain the necessary information. On the basis of the received information was offered the effective approach for the resolving a problem of construction a platform for search, extraction and storage of the information for an estimation of the international scientific and technical activity.

The aim is to develop a platform for searching, retrieving and storing information on a narrow request, and further processing it in order to organize it into a relevant sequence.

The work consists of an introduction, six chapters and conclusions. The introduction reveals the topic of search and use of information in the framework of international scientific and technical cooperation. The first section presents the formulation of the problem of the platform to support decision-making in the field of international scientific and technical cooperation. The second section presents existing decision options, similar to the platform for decision support in the field of international scientific and technical cooperation. The third section offers ways to accomplish this task. The fourth section describes the tools for software development. The fifth section describes the developed software product. The sixth section provides a scenario for using the software product.

Keywords: platform, relevance, criteria, query.

Зміст

Вступ

Вже зараз “виробництво знань” (в широкому сенсі слова) можна назвати однією з фундаментальних галузей будь-якої економіки, оскільки додаткові, унікальні знання, конвертовані в конкретні технології, визначають конкурентоспроможність не тільки окремих товарів або послуг, а й цілих компаній і економік.

Однак, слід зазначити, що “виробництво знань” саме по собі є надзвичайно складним і витратним процесом. Характерною особливістю світового розвитку є посилення впливу “виробництва знань” на всі сторони життя сучасного суспільства; змінюється набір основних факторів і джерел економічного зростання. Під “виробництвом знань” треба розуміти не тільки саму наукову працю, а й весь пласт залучення до цього виробництва: популяризація науки, концентрація уваги медіа-ресурсів на наукових дослідженнях, зміна підходу до опрацювання студентами “крапки входу” до наукових дисциплін, щоб утримувати інтерес потенціальних кандидатів в науковці. Наукові знання перетворюються в інтегральний ресурс, який поряд з природними ресурсами і капіталом, впливає на рівень розвитку держав, а технологічне лідерство стає головною конкурентною перевагою. Науково-технічне співробітництво дозволяє уникнути дублювання зусиль, зайвих витрат, часових втрат, сприяє розумному розподілу суспільного потенціалу, ресурсів і оперативному поширенню інформації; стає реальним інструментом якісної інтеграції будь-якої держави в міжнародний інноваційно-технологічний простір.

Особливість і об'єктивна тенденція сучасного світу така, що прагнення до лідерства в усіх сферах наукових знань, зміцнення зв'язків фундаментальної науки і національних цілей, успішне партнерство держави з науковими та промисловими підприємствами, підготовка висококласних кадрів, при подібному, на перший погляд, наборі напрямків і форм науково-технічного співробітництва, може ставити перед собою різний основний концепт: взаємоузгоджені дії, спрямовані на захист і зміцнення миру (ВООЗ, “Iron Dom”), збереження цивілізації (“Mars Direct”, “Starship”) або глобальне лідерство.

Виходячи з різних концепцій, акцент їх узгодження може змінюватись, але найбільш корисним для цивільного блага є взаємоузгодженість дій держав-виробників знань. Існує велика кількість міжнародних організацій, які об’єднують багато країн з усіх куточків світу під певними галузями науки. Відповідно, процес взаємодії повинен бути узгодженим та налагодженим з метою збереження ефективності співпраці та нарощування КПД досліджень. Вчасно та якісно поєднуючи отримані знання, з-під діяльності організацій можуть виходити проривні та над-ефективні методи, технології, рішення, що постійно допомагають підкорювати локальні та глобальні проблеми людства, покривати потреби.

Взаємоузгодженість дій базується на науковій комунікації, в котрій конкретизують два основних етапи: етап внутрішньої комунікації та етап зовнішньої комунікації. З короткого визначення зовнішньої комунікації можна стверджувати, що вона покриває зв’язок системи науковець-суспільство. В свою чергу внутрішній етап є визначальним в процесі всієї наукової комунікації.

З початкових часів розвитку обміну знанням по-перше мала місце комунікація, заснована на особистих зв’язках між вченими. Це було обумовлено рівнем розвитку засобів передачі інформації, отож простіш за все було спочатку написати листа товаришеві по науковій галузі з сусіднього міста, аніж намагатися поділитися будь-якими результатами з науковцями всіх оточуючих країн та ще й тих, що на іншому континенті. Тому мали місце бути локальні об'єднання вчених (“Республіка вчених”, “Республіка листів”, “Незрима колегія”), де без особистих зусиль була можливість розповсюджувати результати власних досліджень. Таке явище було значно розповсюдженим, навіть знайшло свою гіпотезу про «невидимі коледжі» - самоорганізовані комунікативні об'єднання дослідників, що працюють над новою перспективною проблематикою; групи вчених, що працюють одночасно над одним колом проблем в різних організаціях і країнах; консолідація вчених в «невидимому коледжі» короткочасна: на період вирішення конкретної наукової задачі. З цього можна казати, що такий спосіб взаємозв’язку мав свої недоліки.

Час ніколи не зупиняється, як і розвиток технологій в середовищі комунікації, ближче до ХХ століття з’являються передові засоби передачі інформації свого часу та

колись локальні кола вчених починають розгортатися, інформація передається не тільки з “уст в уста”, а все більш через системи далекого зв’язку і наукове середовище переходить до принципів “відкритої науки”, які ще й досі продовжують формуватися.

Зі зростанням кількості незалежних, відокремлених від університетів, наукових лабораторій, колаборацій (поєднань у співпрацях) інститутів при університетах різних країн, стає вже недостатньо регулярних наукових видань. Суспільство вчених потребує засоби здобуття інформації, які об'єднують в собі швидкість передачі інформації, можливість бути залученим з будь-якого місця планети в будь-який час та повну свободу доступу. Беручи за основу такі потреби, починається підкорення технологій для налагодження комунікації наукової спільноти в мережі Інтернет.

З’явлення наукових журналів та інших засобів обміну інформацією у мережі значно полегшило отримання доступу до свіжих результатів досліджень усіх країн світу, що залучені до однієї чи іншої форми міжнародної співпраці. Маючи на прикметі певні ресурси, можна слідкувати за діяльністю осіб чи структур, що користуються послугами саме цих видань та аналізувати її. Але потреби світового наукового суспільства на цьому не завершувались. Потрібно мати всеосяжний доступ до інформації, бо навіть невеликий висновок маленького інституту може мати великий вплив на глобальні дослідження аналітиків. І як це відбувається за часів стрімко розвиваючихся технологій, потреба знаходить своє рішення - системи пошуку інформації науково-технічного характеру, сама ідея котрих полягає в об’єднанні існуючих ресурсів: видань, журналів, книг, платформ досліджень в один великий простір інформації, що дозволяє користуватися нею, не здійснюючи окремого пошуку кожного ресурсу за одним єдиним запитом.

Найбільш відомими існуючими подібними системами є: Elsevier (колишній Scirus), Google Scholar, CiteSeerX.

Elsevier підтримує власну платформу ScienceDirect (використовує: Scopus - бібліографічна і реферативна база даних та інструмент для відстеження цитованості статей, опублікованих в наукових виданнях; RELX - глобальний постачальник інформаційної аналітики та інструментів прийняття рішень для професіоналів). CiteSeerX індексує наукову літературу та автоматично підраховує індекс цитування

для кількісного визначення значущості окремих публікацій – open-source платформа. Google Scholar - надає особливо велику вагу кількості цитувань.

Здебільшого перевага однієї системи над іншою полягає у наявності платної підписки за користування та доступу до більш розгорнутого змісту публікацій, що, насправді, наразі є приводом для великої кількості суперечок на тему чесної конкуренції пошукових систем наукової інформації та однакових можливостей доступу до цієї інформації всіх діячів науки.

Отож ведучою тенденцією у вирішенні питань комунікації в галузі міжнародного науково-технічного співробітництва є доступність технологій, об’ємність охоплення інформації та визначення її релевантності. Перелік цих факторів визначає актуальність створення системи, що буде вирішувати ці задачі.

1. ПОСТАНОВКА ЗАДАЧІ ПЛАТФОРМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА

Раціональне використання часу, як ресурсу, потребує особливих технічних рішень. Для досягнення певних цілей у дослідженнях аналітику потрібно здобути інформацію, яку він збирається аналізувати в рамках виконання своєї роботи. Процес здобуття інформації має проходити, відносно, швидко, прискорювати прийняття аналітиком рішення, якою саме інформацією скористатись та, таким чином, підтримати прийняття рішення у рамках дослідження.

Платформа для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва повинна представляти собою вирішення питання доступу до наукових робіт національних та комерційних підприємств світу, які вони поширюють у публічно доступних профільних виданнях. В умовах активного дослідження, розробки, вирішення питань людства інструментами виробництва знань кожен учасник процесу повинен мати можливість використовувати релевантну інформацію від досліджень один одного. Взято за основу систему співпраці науковець-науковець, де кожен з учасників не залежить один від іншого матеріально або особово, вони розділені відносно великою дистанцією та різними науковими базами, де проводять роботу. В той час, коли від імені будь-якого учасника публікується матеріал у певному профільному виданні, дослідження або робота, яку проводив учасник, вважається завершеною та публікацію можна використовувати для прийняття рішення за результатами. Задля забезпечення доступу до публікації, економії часу на пошук публікації та впевненості у тому, що публікація дійсно існує на профільному ресурсі, впроваджується платформа для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва.

Необхідно:

1. Проаналізувати існуючі інструментальні засоби та бази даних, які дозволяють отримати необхідну інформацію.
2. Запропонувати ефективний підхід для вирішення задачі побудови платформи для пошуку, вилучення та збереження інформації для оцінки міжнародної науково-технічної діяльності.
3. Розробити алгоритми та програмні засоби для надання аналітику необхідної інформації, впорядкованої за релевантністю відповідно до запиту.

2. ІСНУЮЧІ ВАРІАНТИ РІШЕНЬ, ПОДІБНІ ДО ПЛАТФОРМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА

Наразі, коли технології пошуку та збереження інформації вже здобули свого значного розвитку, існує багато рішень для пошуку інформації у науково-технічній галузі. Одними з провідних можна вважати Scopus, Google Sholar, CiteSeerX, Wolfram Alpha та багато інших.

Найпопулярнішим за інші на поточний час вважається Google Scholar. Google Scholar виконує пошук по наукових публікаціях, серед яких - статті, дисертації, книги, реферати і звіти, опубліковані видавництвами наукової літератури, професійними асоціаціями, вищими навчальними закладами та іншими науковими організаціями. Реєстрація персонального профілю в Google Scholar і робота з ним безкоштовні, ресурс доступний для будь-якого пристрою, підключеного до мережі Інтернет. Однак доступ до пошуку документів не означає доступність повної версії кожного зі знайдених текстів - деякі з них можуть бути завантажені або переглянуті цілком тільки за певну плату, що стягується онлайн-бібліотекамі, що володіють правами на матеріал. Умови доступу до тих чи інших матеріалів визначаються видавцем.

З 2004 року розпочав своє існування Scopus. Одна з найавторитетніших реферативних баз даних вже довгі роки котирується багатьма науковими базами країн світу. Містить в собі короткий опис і відомості про цитування рецензованої літератури: наукових журналів, книг і матеріалів конференцій. Забезпечуючи всебічний огляд результатів світових досліджень в різних областях досліджень, Scopus пропонує інтелектуальні засоби відстеження, аналізу та візуалізації досліджень. Дозволяє виконувати пошук досліджень по коротким описам і вказівниками для вчених. Всі, від дослідників, що прагнуть до наукового прориву, до навчальних закладів, які оцінюють дослідження, можуть використати Scopus в якості

бази даних з короткими описами і покажчиками. У всьому світі база даних Scopus використовується більш ніж у 3000 академічних, державних і корпоративних установах і є їх основним джерелом даних. Зараз Scopus є підрозділом видавництва Elsevier та є здебільшого комерційним продуктом та використовує ресурси Elsevier для доступу до даних.

3. ЗАПРОПОНОВАНІ СПОСОБИ ВИКОНАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Спираючись на формулювання поставленої задачі, основними цілями можна визначити створення рішень пошуку, вилучення та збереження інформації, а також її подальше впорядкування за релевантністю та демонстрація у впорядкованій послідовності для аналітика. З загальних роздумів щодо раціоналізації використання ресурсів при досягненні таких цілей, потрібно поставити підзадачі. Підзадачі дозволять покращити вихідний програмний продукт на етапі розробки та оптимізувати процес розробки. Щоб сформулювати підзадачу потрібно оцінити власні ресурси для розробки та виявити слабкі місця, нівелювати які буде ціль підзадачі.

Розглядаючи основні задачі, найбільший технічний потенціал потребує створення рішень пошуку, вилучення та збереження інформації. Збереження шуканої інформації можливе на носіях, що використовує система. Зазвичай це потребує терабайти місця, адже бази даних, що зберігають науково-технічні публікації та постійно оновлюються, можуть містити сотні тисяч одиниць публікацій. В межах розробки програмного продукту за темою дипломної роботи було прийняте рішення користуватися зовнішнім ресурсом, що має власну базу даних та дозволяє віддалено користуватись єю. А також, що важливо, користуватись безкоштовно. Безкоштовна база даних відкриває можливості для будь-якої взаємодії до всіх охочих, незалежно від роду діяльності, поставлених цілей, залежностей або матеріального достатку. Отже, підзадачею є створення режиму доступу до подібного (віддаленого та безкоштовного) ресурсу, задля економії та раціонального використання внутрішньої пам’яті технічної бази розробки програмного продукту.

Враховуючи те, що обидві задачі є різними за своєю суттю, їх реалізації можна виконати таким чином, щоб була можливість використати реалізацію однієї незалежно від іншої. Беручи поставлені задачі та наведене вище припущення, маємо висновок, що побудова платформи для пошуку, вилучення та збереження інформації може бути реалізованою так, щоб її не потрібно було перелаштовувати при зміні цілей користування інформацією. Тому реалізація впорядкування інформації за

релевантністю та демонстрація у впорядкованій послідовності буде проведена для взаємодії виключно з платформою для пошуку, вилучення та збереження інформації. В той час, як платформа для пошуку, вилучення та збереження інформації може бути використана надалі в інших напрямках розробки для взаємодії з шуканою інформацією. Платформа для пошуку, вилучення та збереження інформації позиціонується як, власне, платформа, на котрій можна використовувати окремо реалізовані засоби опрацювання інформації.

3.1. Побудова платформи для пошуку, вилучення та збереження інформації

Спроможність охоплювати великі обсяги наукових даних потребують використання бази даних, що містить публікації з усіх куточків середовища наукової діяльності в мережі та поєднує їх у собі в вигляді, який їй удобно зчитувати та виводити обрану інформацію. Також потрібно враховувати, що база даних повинна бути віддаленою, надавати можливість безкоштовно користуватися єю. Під користуванням базою даних потрібно розуміти можливість проводити певні дії над нею. Це надсилання запиту з шуканою інформацією до бази даних. Важливо, щоб була можливість робити не звичайний запит, коли пошук виконується по всіх полях бази даних, а більш конкретизований. Це означає, що аналітик, якщо того потребує мета дослідження, повинен мати можливість надіслати вузьконаправлений запит, наприклад, по певних полях бази даних, щоб уточнити вибірку інформації, яку бажано отримати за запитом. Після обробки запиту, повинна повертатись відповідь з результатами запиту. Щоб мати можливість зчитувати будь-які результати, формат збереження даних у базі має бути уніфікованим.

Після проходження етапів пошуку та вилучення даних постає питання їх збереження. Для більш ефективної подальшої обробки інформації потрібен засіб збереження даних, який буде цілком сумісний з мовою програмування та підтримувати формат даних, в якому вони надходять з бази даних разом із відповіддю на запит.

3.2. Впорядкування інформації за релевантністю

Визначаючи термін релевантності можна сказати, що це ступінь відповідності одиниць інформації певним потребам користувача (в межах користування платформою для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва - аналітика). Аналітик потребує одиницю інформації, яка буде мати пріоритет над іншою аналогічною інформацією, що пов’язані суміжним показником. Показником може бути тема публікації, її автор, дата створення публікації, ресурс знаходження публікації тощо. Щоб отримати релевантну впорядкованість в межах будь-якого показника, кожну одиницю інформації потрібно оцінити так, щоб вона мала певним чином відрізнялась від іншої та змогла отримати своє власне місце в послідовності. В джерелах науково-технічної інформації одиницею інформації є публікації. Важілями оцінки публікації виступають декілька критеріїв:

1. Кількість робіт автора за запитом
2. Кількість згадувань автора у БД
3. Кількість осіб, задіяних у публікації (контриб’юторів)
4. Давність публікації

Кількість робіт автора за запитом - кожна тема для дослідження має бути опрацьована багатьма людьми для здобуття найбільш чітких результатів. Адже поєднуючи багато досліджень, зроблених різними дослідниками, виникає більш вірогідна можливість знайти вірне рішення задачі дослідження. Але, якщо, ті ж самі дослідники будуть опрацьовувати ще раз за разом результати попередніх досліджень за темою, тоді якість остаточних результатів, отриманих за весь час роботи з цією темою, буде зростати. Тому існує та використовується такий критерій оцінки публікації, як кількість робіт автора за певною темою.

Кількість згадувань автора у БД - активність наукової діяльності має велике значення у просуванні прогресу. Кожен день проводяться дослідження, вчені роблять нові відкриття, що розширюють горизонти певної наукової галузі. Людина, яка постійно проявляє інтерес до актуальних робіт та приймає в них участь, не тільки підвищує власну цінність, як спеціаліста, але й частіше фігурує у тих чи інших

дослідженнях. Тобто загальна кількість згадувань автора тієї чи іншої публікації у базі даних є одним з свідчень про його професіональної якості.

Кількість осіб, задіяних у публікації (контриб'юторів) - не можна не оцінити вплив колективної роботи над вирішенням проблем людства. Звичайно, є науковці, які своїм особистим інтелектом просували науку, але, цілком, саме поєднання професіоналів дозволяли та досі дозволяють знаходити рішення глобальних проблем та винаходити шляхи покращувати життя людства. З цього можна вивести критерій якості публікації, що кількість науковців, задіяних у роботі над нею, демонструє комплексний та врівноважений підхід до отримання результатів роботи.

Давність публікації - з роками змінюються методи досліджень, трактування попередніх результатів робіт інших вчених. Те, що було визначено багато часу тому, може бути перевизначено, пройшовши повторний аналіз або дослідження, використовуючи нові засоби свого часу та нові результати пов'язаних досліджень. Таким чином, роботи, що були колись опубліковані, можуть втрачати свою актуальність з роками, тому довіри до їх результатів може бути менше, ніж до результатів більш нових досліджень.

Кожен з критеріїв має власне визначаюче значення, чи це кількість авторів або ж те, як багато часу пройшло зі створення публікації. Тому для кожного критерія потрібна нормалізація даних (зведення показників кожного з критеріїв до єдиного вигляду), тоді, поєднуючи нормалізовані значення критеріїв однієї публікації, вона отримує власне значення - рейтинг.

Всі знайдені одиниці інформації повинні бути впорядковані за значенням рейтингу та таким чином створюється релевантна послідовність публікацій. Зберігаючи цю послідовність, знайдена інформація демонструється аналітику. Засобами інтерфейсу позначається релевантність кожної одиниці інформації, що дозволяє аналітику швидше вибирати потрібну інформацію, спираючись на котру приймається рішення в рамках дослідження.

4. Опис засобів розробки

Орієнтуючись на власні технічні можливості, було прийняте рішення використовувати базу даних, складену ресурсом CORE, який дає можливість, використовуючи власний API, зручно звертатися до бази та отримувати будь-яку інформацію, що шукається, за запитом. Збереження отриманої інформації здійснюється засобами pandas. Платформа та додаток розроблені, використовуючи PythonQt з фреймворку Qt у середовищі розробки PyCharm мовою Python. Інтерфейс користувача розроблений у QtCreator.

4.1. Qt Framework

Qt - це багатоплатформний фреймворк для розробки ПЗ на багатьох мовах програмування. Є для Ruby - QtRuby, для Python - PyQt, PHP - PHP-Qt і інших мов програмування. З використанням цього фреймворка написано безліч популярних програм: 2ГІС для Android, Kaspersky Internet Security, Virtual Box, Skype, VLC Media Player, Opera та інші. Qt повністю об'єктно-орієнтована, крос-платформна. Дає можливість розробляти платформо-незалежне ПЗ. Включає в себе безліч класів для роботи з мережею, базами даних, класи-контейнери, а також класи для створення графічного інтерфейсу і безліч інших. Інструментарій розбитий на модулі, кожен з яких розміщується в окремій бібліотеці. Базові класи знаходяться в QtCore, компоненти графічних інтерфейсів - в QtGui, класи для роботи з мережею - в QtNetwork і т.д.

До складу Qt включені інструменти розробника з графічним або консольним інтерфейсом. В тому числі:

assistant - графічний засіб для перегляду гіпертекстової документації по інструментарію та бібліотекам Qt.

designer - графічний засіб для створення і збірки призначених для користувача інтерфейсів на основі компонентів Qt.

qmake - крос-платформний генератор Makefile.

uic - компілятор призначених для користувача інтерфейсів з файлів .ui, створених в Qt Designer.

rcc - компілятор ресурсів з файлів .qrc.

qtconfig - графічний засіб установки налаштувань для додатків Qt.

В Qt використовується CamelCasing: імена класів виглядають як MyClassName, а імена методів - як myMethodName. При цьому імена всіх класів Qt починаються з Q, наприклад QObject, QList або QFont. Більшості класів відповідають заголовки з тим же ім'ям (без розширення .h), тобто використовується:

```
#include <QObject>
#include <QList>
#include <QFont>
```

Для ефективної роботи з класами на стадії виконання в Qt використовується спеціальна об'єктна модель. Багато об'єктів визначаються значенням відразу декількох властивостей, внутрішніми станами і зв'язками з іншими об'єктами. Вони являють собою індивідуальні сутності, і для них не має сенсу операція буквального копіювання, а також поділ даних в пам'яті. В Qt ці об'єкти успадковують властивості QObject.

У тих випадках, коли об'єкт потрібно було б розглядати не як сутність, а як значення (наприклад, при зберіганні в контейнері) - використовуються покажчики. Покажчик на об'єкт, що успадковується від QObject, можно іноді називати об'єктом.

Інші об'єкти (наприклад, контейнери і рядки) повністю визначаються даними, що представлені, тому у відповідних класах є операція присвоювання і конструктор копіювання. Крім того, об'єкти, що представляють однакові дані, можуть розділяти їх в пам'яті.

Ієрархія класів Qt:

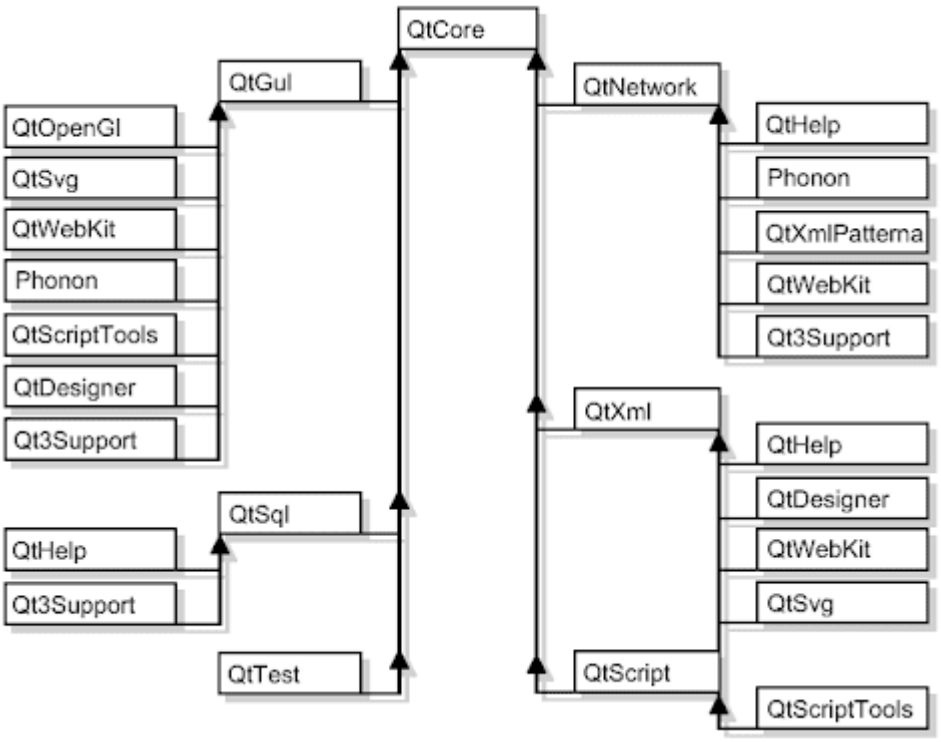


Рисунок 4.1 - Ієрархія класів Qt

Для ефективної роботи з класами на стадії виконання Qt використовує спеціальну об'єктну модель, в якій за допомогою успадкування від QObject і генерування коду компілятором метаоб'єктів реалізовані:

- ієрархії об'єктів;
- система властивостей об'єктів;
- динамічна робота з класами

4.2. PythonQt

PyQt (PythonQt) - це прив'язка Python з відкритим вихідним кодом для віджет-інструментарію Qt, те саме крос-платформне середовище розробки додатків. PyQt складається з більш ніж шести сотень класів, що охоплюють ряд функцій, таких як: графічні інтерфейси, бази даних SQL, веб-інструментарій, обробка XML, мережі.

PyQt доступний в двох редакціях: PyQt4 і PyQt5. PyQt4 надає сполучуючий код для прив'язки версій Qx 4.x і 5.x, в той час як PyQt5 забезпечує прив'язку тільки для

версій 5.x. В результаті PyQt5 не має зворотної сумісності з застарілими модулями більш ранньої версії.

В PyQt доступна велика кількість віджетів для створення додатків за допомогою графічного інтерфейсу. В PyQt5 зроблена реорганізація класів в інші модулі і ревізії в ліцензіях.

Структура каталогів PyQt5:

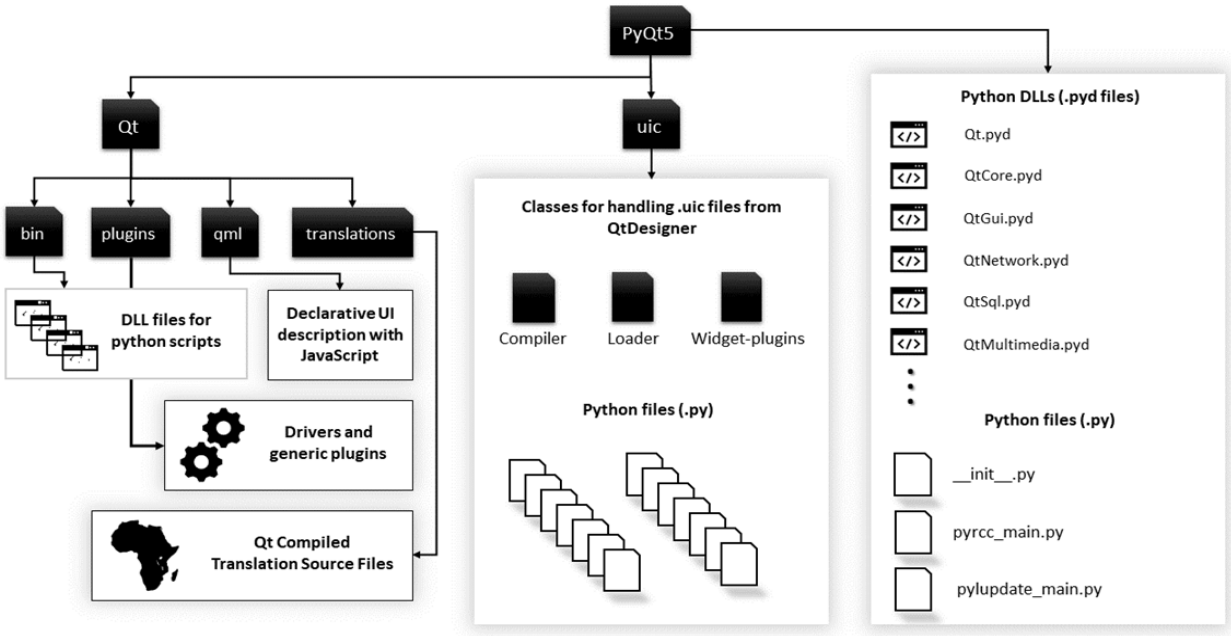


Рисунок 4.2 - Структура каталогів PyQt5

Це фундаментальні модулі, використовувані зв'язуванням Qt в Python, зокрема в PyQt5:

- Qt - об'єднує всі класи / модулі в один. Це значно збільшує обсяг пам'яті, використовуваної додатком. Однак управляти інфраструктурою простіше, імпортуючи тільки один модуль.
- QtCore - містить основні неграфічні класи, використовувані іншими модулями. Тут реалізуються цикл подій Qt, сигнали, можливість підключення до слотів і т. д.
- QtWidgets - містить більшість віджетів, доступних в Pyqt5.
- QtGui - містить компоненти графічного інтерфейсу і розширює модуль QtCore.

- QtNetwork - містить класи, використовувані для реалізації мережевого програмування через Qt. Він підтримує TCP-сервери, TCP-сокети, UDP-сокети, обробку SSL, мережевих сеансів і пошук DNS.
- QtMultimedia - надає низькорівневий мультимедійний функціонал.
- QtSql - реалізує інтеграцію бази даних для баз даних SQL. Підтримує ODBC, MySQL, Oracle, SQLite і PostgreSQL.

4.3. Python

Python простий у використанні, пропонує набагато більше структури і підтримки для великих програм, ніж можуть запропонувати shell-скрипти або пакети файлів. Python також пропонує набагато більше перевірки помилок, ніж C, і, як високорівнева мова, він має вбудовані високорівневі типи даних, такі як гнучкі масиви даних і словники. Через його, більш загальні, типи даних Python застосовується до великої сфері проблем.

Python дозволяє розділити вашу програму на модулі, які можуть бути повторно використані в інших програмах на Python. Він поставляється з великою колекцією стандартних модулів, які можна використовувати в якості основи програм, або як приклади, щоб почати навчання програмуванню на Python. Деякі з цих модулів надають такі речі як файловий ввід / вивід, системні виклики, сокети і інтерфейси інструментарію GUI.

Python - інтерпретована мова, яка може зберегти багато часу в процесі розробки програми, бо немає необхідності в компіляції та компонуванні. Інтерпретатор може використовуватись в інтерактивному режимі, який дозволяє легко експериментувати з особливостями мови, писати одноразові програми або тестувати функції при розробці програм "знизу вгору".

Python дозволяє програмам бути написаними компактно і читаємо. Програми, написані на Python, зазвичай набагато коротше, ніж еквівалентні програми на C, C ++ або Java, з кількох причин:

- високорівневі типи даних дозволяють висловлювати комплекс операцій в одному висловлюванні;
- угруповання висловлювань здійснюється відступом, а не дужками початку і кінця;
- немає необхідності в оголошенні змінних або аргументів.

Python може бути розширеним: легко додати в інтерпретатор вбудовану функцію або модуль як для виконання критичних операцій на максимальній швидкості, так само і зв'язати Python-програму з бібліотеками, які можуть бути доступні тільки в бінарній формі (такі як графічні бібліотеки конкретного виробника).

4.4. CORE

Весь корпус CORE як метаданих, так і повних текстів знаходиться у машинообробному форматі.

CORE надає доступ до 24 936 921 безкоштовних для читання повнотекстових науково-дослідних робіт з 22,419,016 повних текстів, розміщених безпосередньо ресурсом CORE. Це найбільша колекція повних текстів з відкритим доступом, що робить його підходящим інструментом дослідження. Це понад 45 терабайт (ТБ) текстових даних.

Дані CORE можна завантажити як масовий набір даних, що дозволяє обробляти їх на власному комп’ютері або в межах інфраструктури. Набір даних забезпечує узгоджений і збагачений формат даних для доступу до вмісту. Це підходить для створення прототипів нових методів, особливо коли потрібно запускати інтенсивні процеси обробки даних. Це також хороший вибір для аналізу даних та пошуку тексту.

Набір даних в базі створений з інформації, яка була загальнодоступною в Інтернеті. Цей набір даних містить лише вміст відкритого доступу. Включено вміст лише із сховищ та журналів, які занесені до реєстрів, де умовою включення є надання вмісту за ліцензією, сумісною з відкритим доступом. Однак, оскільки метадані часто суперечать, інформація про ліцензії часто не читається машинно, і час від часу в сховища просочується інформація, яка не з відкритим доступом. Набір даних не

призначений для читання оригінальних публікацій, а лише для машинного опрацювання.

Завантажений файл tar-файлу містить стиснуті файли XZ метаданих статті. Стислий файл XZ - це файл з назвою [repositoryID] .json.xz. Після декомпресії кожен рядок текстового файлу містить метадані для 1 статті JSON.

В CORE використаний формат xz завдяки кращому коефіцієнту стиснення проти bzip2 або gzip. Мінусом є те, що інструменти не завжди встановлюються за замовчуванням.

Більшість дистрибутивів Linux має xz, доступний для встановлення в диспетчері пакетів за замовчуванням. Користувачі Mac можуть встановлювати xz через Brew або MacPorts, і є багато інших безкоштовних альтернатив. Користувачі Windows можуть використовувати 7-zip.

Кожен файл JSON не є дійсним JSON, проте кожен рядок є. Кожен рядок розмежовано за допомогою нового формату нової строки Windows (\r\n).

Структура дампа має такий формат:

```
{
  "doi": str|None,
  "coreId": str|None,
  "oai": str|None
  "identifiers": [str],
  "title": str|None,
  "authors": [str],
  "enrichments": {
    "references": [str],
    "documentType": {
       "type": str|None,
       "confidence": str|None
    }
  },
  "contributors": [str],
  "datePublished": str|None,
  "abstract": str|None,
  "downloadUrl": str|None,
  "fullTextIdentifier": str|None,
  "pdfHashValue": str|None,
  "publisher": str|None,
  "rawRecordXml": str|None
  "journals":[str],
  "language": str|None,
  "relations": [str],
  "year": int|None,
  "topics": [str],
  "subjects": [str],
  "fullText": str|None
}
```

CORE збирає, обробляє, гармонізує та збагачує велику кількість метаданих та повних текстів наукових робіт від багатьох постачальників даних. На додаток до цієї постійно зростаючої колекції власний API, що пропонує машинний доступ у режимі реального часу до метаданих та повних текстів наукових робіт, що дозволяє створювати та запускати програми на додаток до вмісту CORE.

4.5. API

Абревіатура API розшифровується як «Application Programming Interface» (інтерфейс програмування додатків, програмний інтерфейс додатку). API розробляється для клієнтів або для внутрішнього використання та обумовлює процес синхронізації інформації між різними програмним забезпеченням. API являє собою сукупність різних інструментів, функцій, реалізованих у вигляді інтерфейсу для створення нових додатків, завдяки якому одна програма буде взаємодіяти з іншою. API визначає можливу функціональність, яку певна програма в форматі бібліотеки або модуля може здійснювати.

Різні компоненти програми завдяки API отримують можливість взаємодії один з одним, формуючи при цьому, як правило, серед компонентів певну систему ранжування, де абсолютно всі компоненти, які вище за рангом використовують інформацію з нижчих за рангом компонентів, які також використовують інформацію з нижчих рангів.

Слід зазначити, що саме за схожим алгоритмом сформовані протоколи передачі даних по мережі Інтернет, де будь-який рівень використовує функціонал нижчого рівня передачі даних, тим самим надаючи необхідну інформацію та функціональні можливості вищому.

JSON api

JSON api - це методологія, згідно з якою ви повинні генерувати запит/відповідь за певними правилами, які встановлює JSON api стандарт. JSON api - це контракт між

клієнтом і сервером про те, як взаємодіяти один з одним; контракт про те, що дані з сервера будуть приходити в JSON форматі.

4.5.1. Окремо про CORE API

CORE API v2 - використовується для доступу до ресурсів, зібраних CORE. API організований за типом ресурсу. Ресурси - це статті, журнали та сховища та представлені у форматі даних JSON. Крім того, кожен ресурс має список методів.

Відповідь на кожен запит містить два поля: status та data. У разі стану помилки поле даних порожнє. Поле даних містить один об'єкт, якщо запит призначений для конкретного ідентифікатора або містить список об'єктів, наприклад для пошукових запитів. У випадку пакетних запитів відповідь - це масив об'єктів, кожен з яких містить власні поля status та data. Для пошукових запитів відповідь містить додаткове поле totalHits, яке є загальною кількістю елементів, що відповідають критеріям пошуку.

Синтаксис пошукового запиту

Складні пошукові запити можуть використовуватися у всіх методах пошуку API. Запит може бути простим рядком або він може бути побудований за допомогою правил та операторів еластичного пошуку.

"Міні-мова" рядка запиту використовується за допомогою Query String Query і параметром рядка запиту q в API пошуку. Рядок запиту аналізується на ряд правил та операторів. Правило може бути одним словом - fast або brown - або фразою, оточеною подвійними лапками - "fast brown" - який шукає всі слова з фрази в такому ж порядку. Оператори дозволяють налаштувати пошук - доступні варіанти пояснюються нижче.

Імена корисних полів - title, description, fullText, authors, publisher, repositories.id, repositories.name, doi, oai, identifiers (що є переліком ідентифікаторів статті, включаючи OAI, URL тощо), language.name та year.

Деякі приклади запитів:

title: psychology AND language.name: English

repositories.id:86 AND year: 2014

4.6. pandas

pandas - це бібліотека з відкритим кодом, що забезпечує високопродуктивні, прості у використанні структури даних та інструменти аналізу даних для мови програмування Python.

Працюючи з табличними даними, такими як дані, що зберігаються в електронних таблицях або базах даних, Pandas зручний інструмент для використання цих даних. Pandas допомагає користуватись, очищати та обробляти дані. У Pandas таблицю даних називають DataFrame.

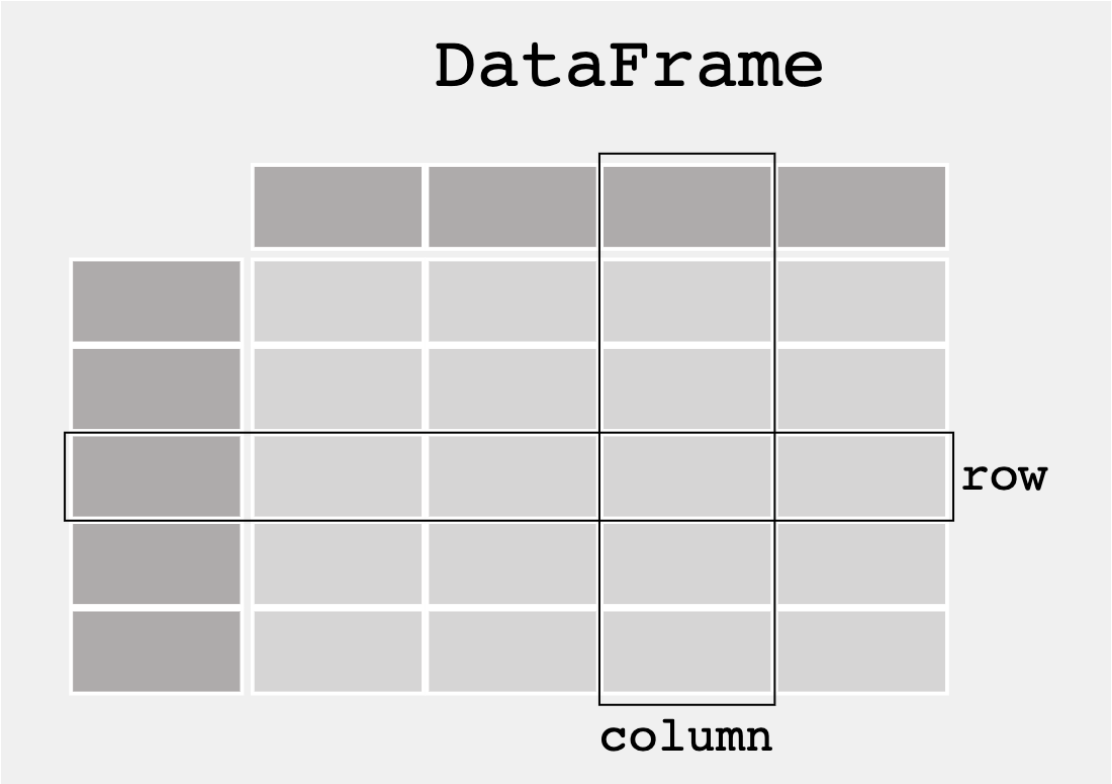
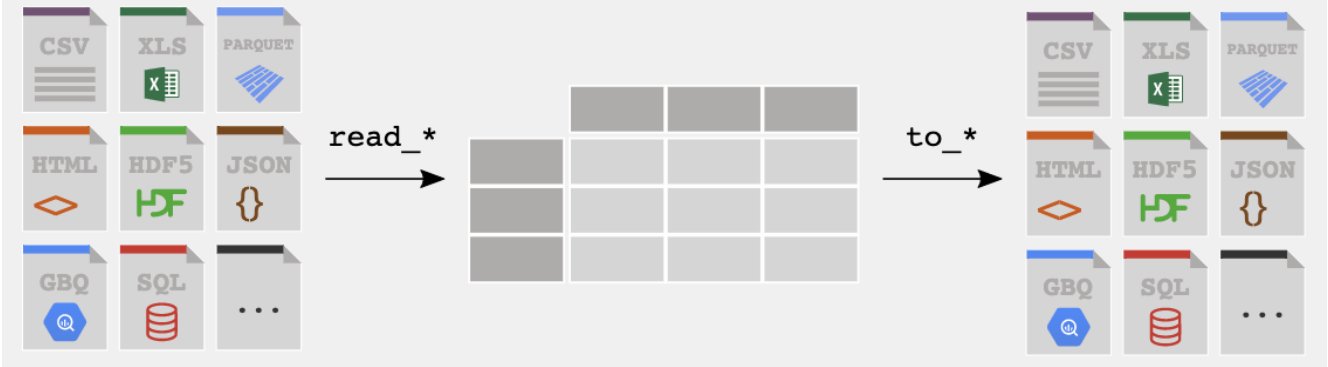


Рисунок 4.3 – таблиця даних DataFrame

Pandas підтримує інтеграцію з багатьма форматами файлів або джерелами даних (csv, excel, sql, json, parket,…). Імпорт даних з кожного з цих джерел даних



забезпечується функцією з префіксом read_ *. Аналогічно, метод to_ * використовується для експорту даних.

Рисунок 4.4 – загальна схема імпорту-експорту даних у DataFrame

5. Опис програмного продукту

Беручи за основу запропонований спосіб поставлення задачі, визначаємо, що програмний продукт складається з двох основних частин. Це платформа для пошуку, вилучення та збереження інформації, та додаток, що впорядковує інформацію за релевантністю.

Наведена нижче схема демонструє схему взаємодії платформи для пошуку, вилучення та збереження інформації з додатком, що впорядковує інформацію за релевантністю.

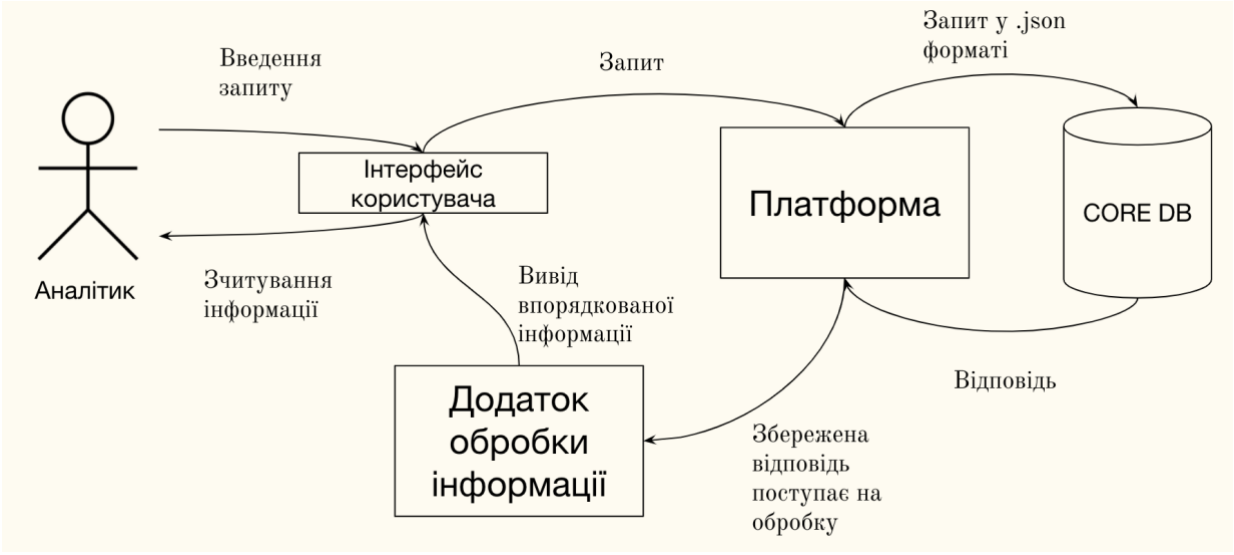


Рисунок 5.1 – Схема програмного продукту

Модулями програмного продукту є: Платформа, що виконує функцію пошуку, отримання та збереження інформації; Додаток обробки інформації, що виконує функцію впорядкування знайденої інформації за релевантністю; Інтерфейс користувача, що надає можливість вводити запит та отримувати, впорядковану за релевантністю, відповідь.

5.1. Платформа для пошуку, вилучення та збереження інформації

Ресурсом, що надає доступ до інформації, було обрано базу даних CORE. Вибір використовувати саме цей ресурс був обумовлений певними факторами.

По перше, CORE забезпечує пошуковий доступ до колекції з понад 125 мільйонів результатів досліджень з відкритого доступу. Всі результати доступні для безкоштовного завантаження. Колекція зібраних результатів доступна шляхом перегляду останніх доповнень або шляхом доступу через власний API. Отож, використовуючи CORE, вирішуються задачі економії власного технічного ресурсу у вигляді великоємного носія інформації, а також нивелювання грошових витрат, адже більшість бібліографічних баз даних стягують певну плату за користування власними ресурсами. CORE також може надавати платні послуги, але вони вже стосуються безпосередньо швидкості отримання відповіді, в той час, як доступ до користування даними залишається безкоштовним. Названі аспекти забезпечують перспективність використання платформи, що виконує функцію пошуку, отримання та збереження інформації, тому що не потрібно поновлювати контракти на доступ до бази даних та не потрібно забезпечувати розширення ємності внутрішньої пам’яті технічного засобу, на якому використовується платформа, тому що CORE надає віддалений доступ до своєї бази даних через власний API.

Для програмного продукту було розроблено пакет модулів мовою Python, що забезпечують функціонування платформи для пошуку, вилучення та збереження інформації.

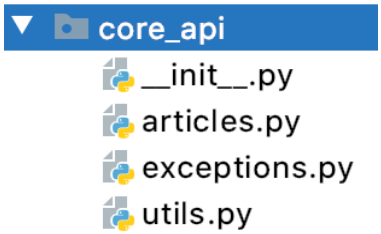
Модуль - це файл, який містить визначення і оператори Python. Ім’я файлу є ім'ям модуля з доданим суфіксом .py. Разом з визначеннями функцій модулі можуть містити виконавчі оператори. Ці оператори призначені для ініціалізації модуля. Вони виконуються лише один раз, коли в операторі імпорту зустрічається ім'я модуля (вони також запускаються, якщо файл виконується як скрипт). Кожен модуль має власну

приватну таблицю імен, яка використовується як глобальна всіма функціями, визначеними в модулі. Модулі можуть імпортувати інші модулі. Прийнято розміщувати всі оператори import на початку модуля. Ім'я імпортованого модуля поміщається в глобальну таблицю імен імпортуючого модуля. Для підвищення ефективності кожен модуль імпортується тільки один раз за сесію інтерпретатора.

Пакети є способом структурування простору імен Python за допомогою "точкових імен модулів". Наприклад, ім'я модуля A.B позначає подмодуль з ім'ям B в пакеті з ім'ям A.

5.1.1. Пакет core_api для пошуку та вилучення інформації

Пакет, що відповідає за надсилання запитів до бази даних CORE через CORE API, має назву core_api та складається чотирьох модулів:



__init__.py - модуль ініціалізації. Файли __init__.py необхідні, щоб Python трактував каталоги як пакети; це робиться для того, щоб каталоги із загальною назвою не приховували дійсні модулі, які з’являються пізніше на шляху пошуку модуля.

В випадку використання CORE API __init__.py зчитує спецальний ключ доступу, який надається CORE, та перевіряє його за дозволеними CORE ключами, використовуючи функцію update_api_key() з модулю utils.py. Щоб отримати дозвіл на користування базою даних через CORE API, потрібно зареєструватись та отримати особовий ключ доступу, який потрібно зберігати в файлі JSON, як config.json.

Імпортована до __init__.py з модулю utils.py функція update_api_key()

```
def update_api_key(api_key):
    global g_api_key
    g_api_key = api_key
```

Перевірка відповідності ключа у __init.py__

```
with open('config.json') as config_file:
    json_data = json.load(config_file)

    if 'CORE_API_KEY' in json_data:
        update_api_key(json_data['CORE_API_KEY'])
```

exceptions.py - модуль визначення класів виключень, які повинні спрацьовувати при певних помилках.

utils.py - модуль, що відповідає за безпосередній зв'язок з базою даних CORE через CORE API. До модулю utils.py імпортовані простіри імен requests, певні класи модулю exceptions, logging та time.

```
import requests
from .exceptions import GeneralHTTPException, MalformedJSONRequest, InvalidAPIKey, \
                        TooManyQueriesInRequest, TooManyRequests, ProxyError, ResponseTimeout, \
                        PageNotFound, InternalServerError

import logging
import time
```

Бібліотека запитів Python - requests - дозволяє надсилати запити HTTP в Python. Оскільки при використанні API відправляються запити HTTP і отримуються відповіді, бібліотека Requests відкриває можливість використання API в Python.

Запити HTTP лежать в основі всесвітньої мережі. Браузер направляє запит на сервер веб-сторінки. Сервер відповідає на них, пересилаючи всі необхідні дані для виведення сторінки, і браузер відображає сторінку, щоб її можна було побачити. Цілком цей процес виглядає так: клієнт (наприклад браузер або скрипт Python, що використовує бібліотеку Requests) відправляє дані на URL, а сервер з цим URL зчитує дані, вирішує, що з ними робити, і відправляє клієнту відповідь. Після цього клієнт може вирішити, що робити з отриманими у відповіді даними.

У складі запиту клієнт відправляє дані по методу запиту. Найбільш поширеними методами запиту є GET, POST та PUT. Запити GET зазвичай призначені тільки для зчитування даних без їх зміни, а запити POST і PUT зазвичай призначаються для зміни даних на сервері.

В реалізації платформи для пошуку, вилучення та збереження інформації достатньо використання GET запитів, адже необхідно лише отримати інформацію для користування аналітиком.

Запит реалізовано у функції api_request().

```
response = requests.get(url, params=params, timeout=60)
```

Аргументами методу .get() виступають посилання на ресурс та params, значення якого дорівнює значенню ключа.

```
params['apiKey'] = g_api_key
```

По перше в якості відповіді ми отримуємо код стану, який свідчить про успіх знаходження інформації за запитом .get(). Існує деяка кількість кодів стану, котрі в свою чергу свідчать про певні помилки, які можуть виникнути при обробці запиту сервером.

Якщо ми отримаємо відповідь у вигляді коду стану ‘200’, тоді одразу за ним ми отримаємо відповідь від сервера з результатом обробки запиту.

```
if response.status_code == 200:
    return response.json()
```

Модуль exceptions визначає класи виключень, які містять базові та користувацькі (розроблені окремо) виключення. Виключення спрацьовують при виникненні помилок в процесі роботи із запитом. В тому числі з кодами стану під час помилок при обробці запиту на сервері. Наприклад:

```
elif response.status_code == 401:
    logging.error('[core api] Got InvalidAPIKey status code in response!')
    raise InvalidAPIKey
elif response.status_code == 403:
    logging.error('[core api] Got TooManyQueriesInRequest status code in response!')
    raise TooManyQueriesInRequest
elif response.status_code == 429:
    logging.error('[core api] Got TooManyRequests status code in response!')
    raise TooManyRequests
```

Такі виключення виникають при помилці в ключі доступу до CORE API, коли забагато введених запитів у запиті до сервера або забагато запитів надіслано до серверу за певний проміжок часу.

Бібліотека logging - дуже корисний інструмент для програміста. Вона допомагає краще зрозуміти потік програми та виявити сценарії. Logging українською можна описати як “ведення журналу”, де Logs - це журнали.

Журнали надають розробникам додатковий набір очей, які постійно переглядають потік, через який проходить додаток. Вони можуть зберігати інформацію, наприклад, який користувач чи IP доступ звернувся до програми. Якщо виникла помилка, вони можуть надати більше уявлень, ніж трек стека, повідомляючи, яким був стан програми до того, як вона надійшла до рядка коду, де сталася помилка.

Записуючи у журнал корисні дані з потрібних місць, можна легко відладжувати помилки, використовувати дані для аналізу продуктивності програми для планування масштабування або перегляду моделей використання.

Python надає систему ведення журналів як частину його стандартної бібліотеки, тому можна швидко додати журнал до своєї програми, імпортувавши бібліотеку logging.

Імпортуючи модуль logging, ви можете використовувати "logs" (логи) для ведення журналу повідомлень, які ви хочете бачити. За замовчуванням є 5 стандартних рівнів, що вказують на подію. Кожен має відповідний метод, який можна використовувати для реєстрації події на відповідному рівні. Визначеними рівнями, в порядку збільшення ступеня вираженості, є наступні:

- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

Модуль logging надає вам логер за замовчуванням, який дозволяє вам розпочати роботу, не потребуючи окремої конфігурації. Відповідні методи для кожного рівня визначаються як:

```
import logging

logging.debug('This is a debug message')
logging.info('This is an info message')
logging.warning('This is a warning message')
logging.error('This is an error message')
logging.critical('This is a critical message')
```

Для ведення журналу стану посеред виконання методів модулю utils.py використовуються методи логу `logging.info('This is an info message')`, наприклад:

```
response = requests.get(url, params=params, timeout=60)
logging.info('[core api] Try n: {n} completed in: \'{time:.3}\' sec'
             .format(n = request_try + 1, time = time.time() - time_before))
```

Лог фіксує інформацію про кількість спроб запиту до серверу та час надходження запиту.

```
elif response.status_code == 401:
    logging.error('[core api] Got InvalidAPIKey status code in response!')
    raise InvalidAPIKey
```

Лог фіксує виникнення помилки.

Логи зберігаються у каталозі логів у кореневому каталозі програмного продукту.

Бібліотека time надає методи для операціями з часом, особливо це використовується в реалізації логів (logs) бібліотеки logging.

articles.py - модуль, в якому реалізовано отримання ідентифікаторів публікацій, які були знайдені відповідно до запиту.

До модулю articles.py імпортовані функція api_request() з модулю utils.py, клас виключень GeneralHTTPException та простір імен з логгінгу.

```
from .utils import api_request
from .exceptions import GeneralHTTPException
import logging
```

Для отримання ідентифікаторів знайдених публікацій у функції search() спочатку розраховується кількість співпадінь за запитом та привласнюється до змінної hits.

```
hits = 0
try:
    hits = api_request('articles/search/{query}'.format(query=query), {})['totalHits']
except GeneralHTTPException:
    logging.error('[core api] GeneralHTTPException occurred while retrieving hits')
```

Скориставшись функцією api_request() з модулю utils.py отримуємо кількість співпадінь за запитом.

Дані в БД CORE зберігаються та формуються у сторінках в кількості 100 одиниць на сторінку, тому потрібно розуміти, скільки максимум сторінок даних нам потрібно вилучити, спираючись на кількість співпадінь.

```
max_pages = int(hits / 100) + 1
pages = 10 if max_pages > 10 else max_pages
```

В межах отриманої кількості сторінок отримуємо та зберігаємо інформацію про знайдені публікації в контейнер articles.

```
articles = api_request('articles/search/{query}'.format(query=query),
                       {'pageSize': 100, 'page': page + 1,
```

Та накопляємо ідентифікатори знайдених публікацій в ids[].

```
for rec in articles['data']:
    ids.append(rec['id'])
```

ids[] повертається як значення функції search()

5.2. Додаток обробки інформації

Додаток може бути опційним доповненням платформи для пошуку, вилучення та збереження інформації. Реалізований додаток виконує задачу з впорядкування знайденої платформою інформації за релевантністю.

Релевантність - це відповідність результату очікуванням. В контексті пошукової видачі релевантність демонструє, наскільки точний розгорнуту відповідь користувач отримав на свій запит.

У результатах пошуку найбільш релевантні сторінки розташовані нагорі. Тобто, чим краще, на думку пошукової системи, ресурс відповідає запиту користувача - тим більш високу позицію він займає.

Релевантність - один з головних показників ефективності роботи пошукової системи, який визначається за спеціальним алгоритмом.

Визначити релевантність можливо за певними критеріями, по яких оцінюється публікація або будь-яка інша одиниця інформації. Для оцінки релевантності публікацій, вилучених за запитом з БД CORE запропоновано наступні критерії:

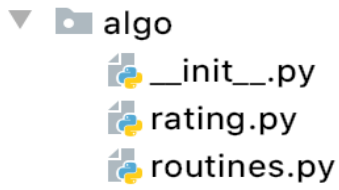
1. Кількість робіт автора за запитом
2. Кількість згадувань автора у БД
3. Кількість осіб, задіяних у публікації (контриб’юторів)
4. Давність публікації

Список критеріїв оцінювання може бути розширений відповідно до додаткових вимог аналітика.

За метою задачі було створено алгоритм, котрий розраховує релевантність кожної публікації відносно іншої та впорядковує їх в послідовності від публікації з найвищим значенням релевантності до найнижчого.

5.2.1. Пакет algo збереження та обробки інформації

Додаток обробки інформації реалізовано у пакеті algo, що складається з трьох модулів:



В даному випадку __init__.py - модуль ініціалізації. Він не містить програмного коду, адже необхідний лише для того, щоб Python трактував каталог algo як пакет.

Модуль routines.py - своєрідний режим доступу алгоритму визначення релевантності до публікацій. До модулю імпортовані наступні об’єкти:

```
import logging
import core_api.articles
import json
import pandas as pd
import re
import os.path
from core_api.exceptions import GeneralHTTPException
```

Логгінг для фіксації сценаріїв, контейнер з даними знайдених публікацій, бібліотека JSON, простір імен використання pandas, модуль з методами роботи з регулярними виразами re, модуль .path бібліотеки os, а також клас виключень GeneralHTTPException.

JSON - стандартизований формат, який використовується для передачі даних. Python містить вбудований модуль під назвою json для кодування і декодування даних JSON. Модуль json надає метод dump () для запису даних в файли. Також є метод dumps () для записів в рядок Python. dump () приймає два позиційних аргументи: об'єкт даних, який серіалізуются (зводиться до байтового запису) і, файловий об'єкт, в який будуть вписані байти.

З опису засобів розробки про pandas - це інструмент для використання даних з таблиць або баз.

Регулярний вираз - це послідовність символів, яка використовується для пошуку і заміни тексту в рядку або файлі. Модуль re призначений для роботи з регулярними виразами мовою Python.

Найчастіше регулярні вирази використовуються для:

- пошуку в рядку
- розбиття рядка на підрядки
- заміни частини рядка

os.path - модуль, який реалізує деякі корисні функції роботи з шляхами та є вкладеним в модуль os.

Модуль routines.py розміщує дані, отримані за ідентифікаторами знайдених за запитом публікацій у фрейми збереження інформації pandas DataFrames.

```
articles = []
for index, id in enumerate(ids[:20]):
    logging.info('[algo] Fetching article id: \'{id}\''.format(id=id))
    statusSignal.emit('Fetching article id: \'{id}\''.format(id=id))

    try:
        data = core_api.articles.get(id)
```

Збираємо дані потрібних публікацій у data, використовуючи функцію get() модулю articles пакету core_api.

```
def get(id):
    return api_request('articles/get/{id}'.format(id=id), {})['data']
```

Отримані дані з data збираються у контейнері articles.

```
articles.append(data)
```

Метод dumps() записує дані articles у .json.

```
with open(json_data_dump_path.format(query=query), 'w') as dump_file:
    dump_file.write(json.dumps(articles))

articles_df = pd.DataFrame(articles)
articles_df.set_index('id', inplace=True)

articles_df.to_pickle(pkl_dump_path.format(query=query))

return articles_df
```

Засобами pandas всі отримані дані публікацій розміщені у форматі .pkl та повертаємо articles_df повертається, як значення функції find_and_fetch_articles().

Модуль rating.py здійснює обробку публікації за критеріями. До модулю імпортовані модуль articles пакету core_api, бібліотека logging та засоби роботи з pandas.

Функція author_hits() відповідає за підрахування кількості робіт автора за запитом та кількості згадувань автора у БД.

В модулі articles пакету core_api виконується запит та отримується відповідь з кількістю повторень автора поточної публікації у БД CORE. Ця дія справедлива для всіх авторів публікації.

```
try:
    response = api_request('articles/search/author:\"{author}\"'.format(author=author), {})
    logging.info('[core api] Got {n} author hits'.format(n=response['totalHits']))
    return response['totalHits']
```

В функції author_hits() контейнер core_hits[] заповнюється поверненим результатом запиту на кількість повторень автора у БД CORE.

```
core_hits = []
for index, author in enumerate(authors):
    statusSignal.emit('Retrieving rating for author \'{author}\''.format(author = author))
    core_hits.append(core_api.articles.get_author_hits_n(author))
    progressSignal.emit(index, len(authors))
```

В свою чергу контейнер self_hits[] призначений, щоб містити кількість робіт автора серед повернених за результатом запиту. Якщо ім’я автора поточної публікації

зустрічає повторення серед авторів інших отриманих публікацій, тоді значення змінної hit зростає.

```
self_hits = []
for author in authors:
    hit = 0

        if author == raw_author: hit += 1

    self_hits.append(hit)
```

У контейнері self_hits[] розміщуються кількості робіт автора серед повернених за результатом запиту.

Функція rate() модулю rating.py оброблює публікації за двома останніми критеріями, що були запропоновані для реалізації додатку обробки інформації, та поєднує усі отриманні нормалізовані значення, щоб привласнити до публікації значення, котре надасть можливість впорядкувати публікації за релевантністю.

Зведення двох критеріїв - кількості повторень автора у БД CORE та кількості повторень автора серед авторів отриманих за запитом публікацій - до нормалізованого виду.

Нормалізація даних дозволить уніфікувати отримані результати обробки, щоб їх можно було співставляти один до одного.

Композиція контейнерів self_hits[] та core_hits[] у authors_rates та зведення значень authors_rates до нормалізованого виду шляхом отримання відношення кожного значення у author_rates до максимального.

Обробка та зведення до нормалізованого виду за критеріями давності публікації та кількості осіб, що приймали участь в створенні публікації (контриб'юторів).

```
rates['year'] = articles['year'].map(lambda y: 0 if pd.isna(y) else y / articles['year'].max())

rates['contributors'] = articles['contributors'].map(lambda arr: len(arr)) / \
                        (max(articles['contributors'].map(lambda arr: len(arr))) + 1)
```

Композиція всіх результатів обробки у reduced_rates та повернення у функції rate() нормалізованої спадаючої послідовності.

```
reduced_rates = rates.apply(lambda r: r['year']*2 + r['contributors'] + r['authors']*3, axis = 1)

return reduced_rates.apply(lambda r: r / reduced_rates.max()).sort_values(ascending=False)
```

5.3. Інтерфейс користувача

Платформа для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва має модуль інтерфейсу користувача. Інтерфейс користувача реалізовано таким чином, щоб аналітик мав можливість швидко виконати запит та отримати, впорядковану за релевантністю, шукану інформацію.

Інтерфейс складається з поєднання трьох елементів:

- елементу відображення поточного стану роботи програмного продукту;
- вікна виведення інформації;
- рядка введення запиту.

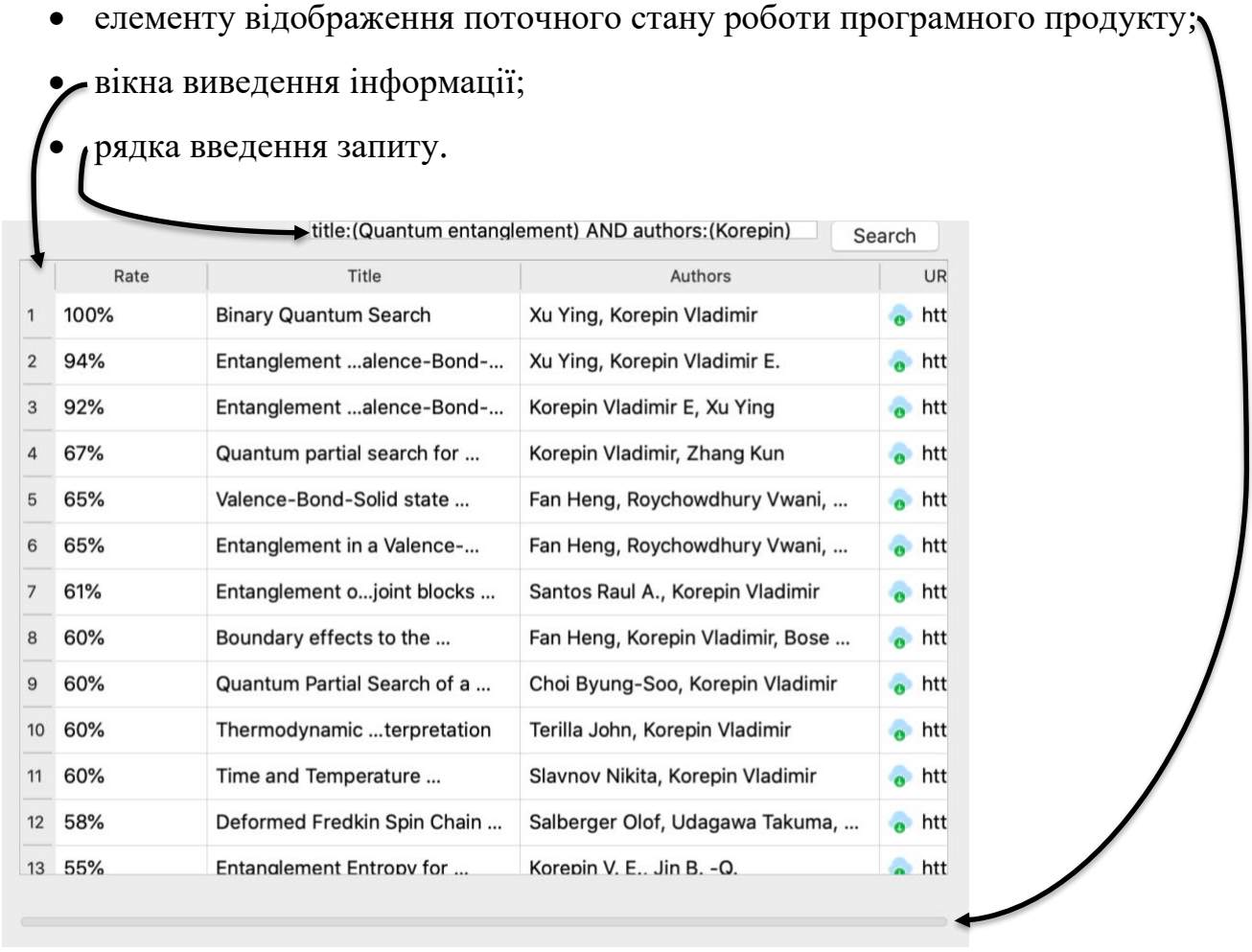


Рисунок 5.2 – інтерфейс користувача

Результат введення запиту до пошукової строки - виведення послідовності впорядкованих за релевантністю публікацій.

Колонка Rate містить процентне відображення релевантності відносно інших публікацій. Колонка Title містить назви публікацій. Колонка Authors містить авторів публікації. Колонка URL містить режим доступу до публікації у форматі .pdf.

В рядок запиту можна вводити запит з використанням синтаксису еластичного пошуку.

"Міні-мова" рядка запиту використовується за допомогою Query String Query. Рядок запиту аналізується на ряд правил та операторів. Правило може бути одним словом - fast або brown - або фразою, оточеною подвійними лапками - "fast brown" - який шукає всі слова з фрази в такому ж порядку. Оператори дозволяють налаштувати пошук - доступні варіанти пояснюються нижче.

Імена корисних полів - title, authors, publisher, repositories.id, repositories.name, identifiers (що є переліком ідентифікаторів статті), language.name та year.

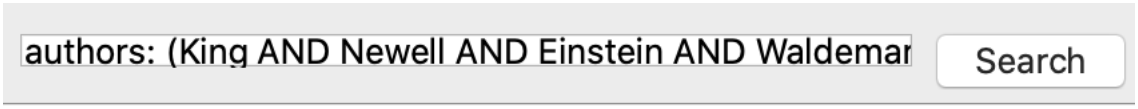
Деякі приклади запитів:

- Стаття Psychology англійською мовою:

title: psychology AND language.name: English

- Перелік прізвищ авторів для знайдення їх спільних робіт:

authors: (King AND Newell AND Einstein AND Waldemar)



6. Сценарій використання

Платформа для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва створена для оцінки діяльності міжнародного співробітництва в науково-технічній галузі. Аналітик має можливість отримати релевантну інформацію, що має можливість підтримувати рішення аналітика в процесі оцінки або дослідження.

Якщо аналітику потрібно оцінити діяльність інституту однієї країни у порівнянні з інститутом іншої країни стосовно опрацювання певної теми з науково-технічної галузі, то він має можливість скористатись синтаксисом еластичного пошуку та сформувати два запити, порівнюючи результати котрих можна робити висновки в межах проведення оцінки або дослідження.

Запропоновано до першого запиту здійснити пошук релевантних публікацій на тему Quantum entanglement (Квантова заплутаність), де автором робіт за такою темою виступає науковець на прізвище Korepin (Корепін).

Оператори title та authors визначають, відповідно, назву роботи та автора. Умови запиту поєднані логічним І (AND).

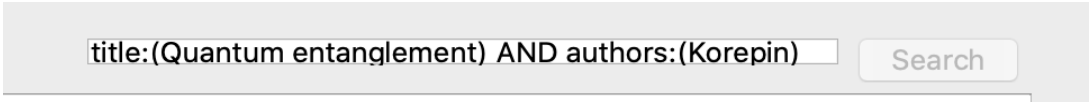


Рисунок 6.1 – пошуковий рядок

Елемент відображення поточного стану роботи свідчить про те, що зараз йде отримання даних з 3 сторінки та який стан процесу.

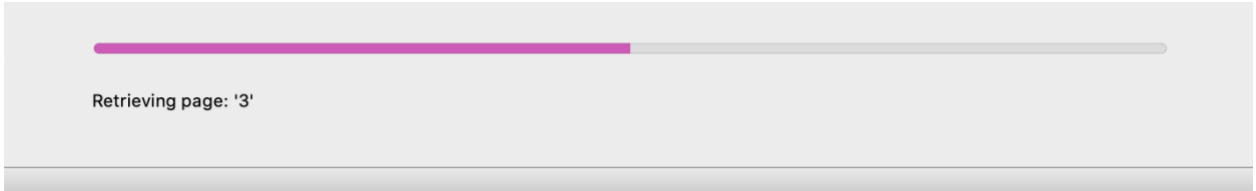


Рисунок 6.2 – поточний стан роботи програмного продукту

Через деякий час тому ж елементі видно, що зараз відбувається процес отримання рейтингу для автора публікації, що проходить аналіз.

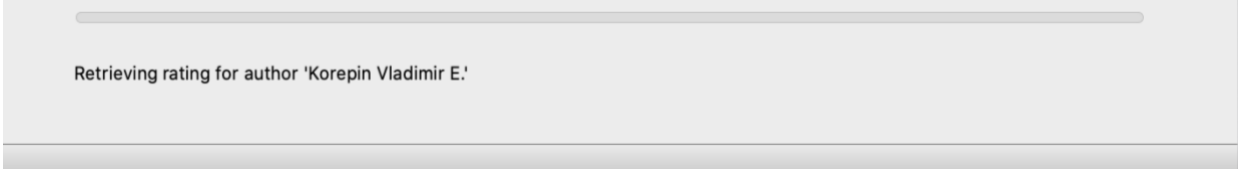
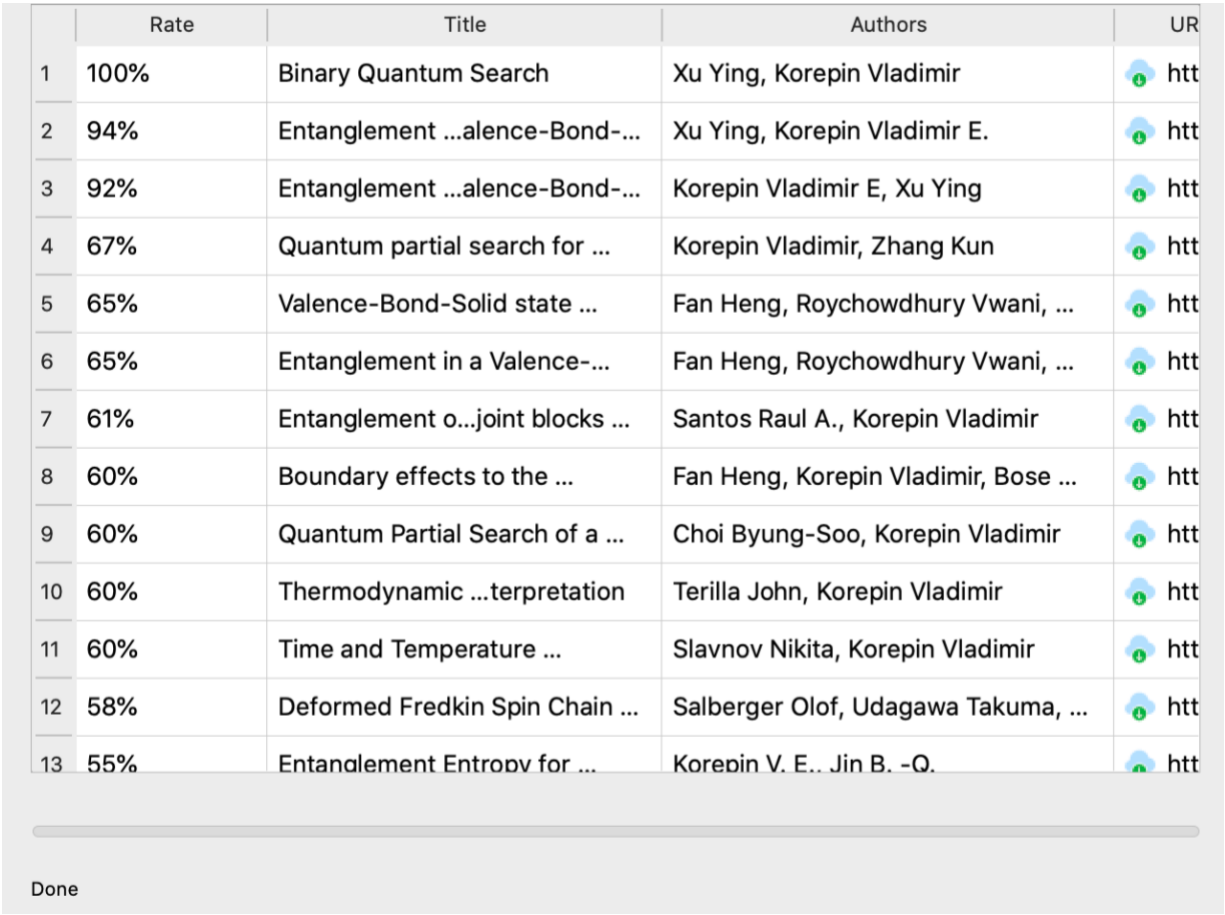


Рисунок 6.3 – поточний стан роботи програмного продукту

Остаточний результат пошуку має наступний вигляд



	Rate	Title	Authors	UR
1	100%	Binary Quantum Search	Xu Ying, Korepin Vladimir	htt
2	94%	Entanglement ...alence-Bond-...	Xu Ying, Korepin Vladimir E.	htt
3	92%	Entanglement ...alence-Bond-...	Korepin Vladimir E, Xu Ying	htt
4	67%	Quantum partial search for ...	Korepin Vladimir, Zhang Kun	htt
5	65%	Valence-Bond-Solid state ...	Fan Heng, Roychowdhury Vwani, ...	htt
6	65%	Entanglement in a Valence-...	Fan Heng, Roychowdhury Vwani, ...	htt
7	61%	Entanglement o...joint blocks ...	Santos Raul A., Korepin Vladimir	htt
8	60%	Boundary effects to the ...	Fan Heng, Korepin Vladimir, Bose ...	htt
9	60%	Quantum Partial Search of a ...	Choi Byung-Soo, Korepin Vladimir	htt
10	60%	Thermodynamic ...terpretation	Terilla John, Korepin Vladimir	htt
11	60%	Time and Temperature ...	Slavnov Nikita, Korepin Vladimir	htt
12	58%	Deformed Fredkin Spin Chain ...	Salberger Olof, Udagawa Takuma, ...	htt
13	55%	Entanglement Entropy for ...	Korepin V. E., Jin B.-Q.	htt

Рисунок 6.4 – результат пошуку

Як видно, результат свідчить про те, що є велика вибірка робіт за участі автора Володимира Корепіна за темою Квантової заплутаності з різною релевантністю.

Другий пошуковий запит буде виконаний за такою ж самою темою, але авторства науковця з іншої країни. Наприклад, за участі науковця з прізвищем Formaggio (Формаджо).

Результат запиту має наступний вигляд.

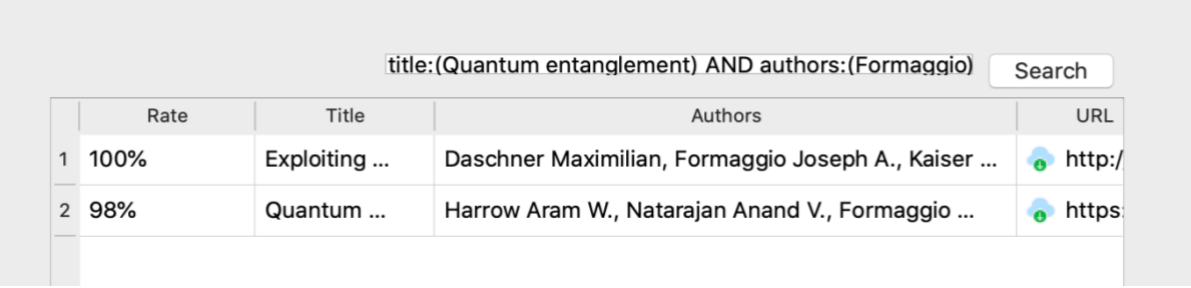


Рисунок 6.5 – результат пошуку

Можна стверджувати, що кількість робіт за участі пана Формаджо відрізняється від кількості робіт пана Корепіна, поєднаних одною темою, отже відрізняється й зануреність та активність у темі інститутів, які можуть представляти ці науковці.

Висновки

1. В процесі досягнення мети дипломної роботи було проаналізовано існуючі інструментальні засоби та бази даних, які дозволяють отримати необхідну інформацію.

2. Запропоновано ефективний підхід для вирішення задачі побудови платформи для пошуку, вилучення та збереження інформації для оцінки міжнародної науково-технічної діяльності.

3. Розроблено алгоритми та програмні засоби для надання аналітику необхідної інформації, впорядкованої за релевантністю відповідно до запиту.

Розроблена платформа для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва має перспективну можливість допомагати спеціалістам в науково-технічній галузі здійснювати прогрес у дослідженнях, які можуть вирішувати глобальні проблеми та питання комфорту життя. Розроблений засіб та подібні до розробленого, вже існуючі, засоби для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва за умов належного використання, розширення функціональних можливостей та швидкості опрацювання запитів в перспективі можуть сприяти науковцям в їх роботі з покращення життя людства.

Список використаних джерел

1. Qt Framework - Qt Features / – Режим доступу до ресурсу: https://www.qt.io/product/features
2. Python 3 и PyQt 5. Разработка приложений. 2-е издание / Прохоренок, Дронов
3. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces 1st Edition / Mark Masse
4. Обзор HTTP / С.М. Тимачев

Python for Data Analysis. Data Wrangling with Pandas, NumPy, and Ipython / Wes McKinney

5. Learning pandas / Michael Heydt

Continuous API Management / Mehdi Medjaoui, Erik Wilde, Ronnie Mitra, Mike Amundsen

6. Python API Development Fundamentals / Jack Chan, Ray Chung, Jack Huang
7. Building RESTful Python Web Service / Gaston C. Hillar

Додаток 1

ПЛАТФОРМА ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА

Специфікація

УКР.НТУУ“КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТР-62137_20Б

Аркушів 3

Київ – 2020

Позначення	***Найменування***	***Примітки***
	Документація	
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТР-62137_20Б	Записка.docx	Пояснювальна записка
	Компоненти	
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ ТР-62137_20Б_12-1	__init__.py articles.py exceptions.py utils.py	Файли програмного продукту, що містять реалізацію пошуку та вилучення інформації
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ ТР-62137_20Б_12-2	__init__.py rating.py routines.py	Файли програмного продукту, що містять реалізацію збереження та впорядкування інформації в релевантну послідовність
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ ТР-62137_20Б_12-3	__init__.py handlers.py utils.py	Файли програмного продукту, що містять реалізацію інтерфейсу користувача

УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР-62137_20Б_12-4	main.py	Файл програмного продукту, що відповідає за запуск інтерфейсу користувача

Додаток 2

ПЛАТФОРМА ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА

Текст програми

УКР.НТУУ“КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР-62137_20Б_12-1

УКР.НТУУ“КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР-62137_20Б_12-2

УКР.НТУУ“КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР-62137_20Б_12-3

УКР.НТУУ“КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР-62137_20Б_12-4

Аркушів 11

Київ – 2020

```
import json
from .utils import update_api_key
from .exceptions import NoApiKeyInConfigFile

__all__ = ['articles', 'exceptions']

with open('config.json') as config_file:
    json_data = json.load(config_file)

    if 'CORE_API_KEY' in json_data:
        update_api_key(json_data['CORE_API_KEY'])
    else:
        raise NoApiKeyInConfigFile


from .utils import api_request
from .exceptions import GeneralHTTPException
import logging


def get_author_hits_n(author):
    logging.info('[core api] Retrieving author hits:
\'{author}\''.format(author=author))

    try:
        response =
api_request('articles/search/author:\"{author}\"'.format(author=author)
, {})
        logging.info('[core api] Got {n} author
hits'.format(n=response['totalHits']))
        return response['totalHits']

    except GeneralHTTPException:
        logging.error('[core api] GeneralHTTPException occurred while
retrieving author hits: \'{author}\''.format(author=author))

    return 0

def search(query, progressSignal, statusSignal):
    logging.info('[core api] Searching query: \'{query}\''.format(query
= query))

    hits = 0
    try:
        hits =
api_request('articles/search/{query}'.format(query=query),
{})['totalHits']
    except GeneralHTTPException:
        logging.error('[core api] GeneralHTTPException occurred while
retrieving hits')
```

```
    logging.info('[core api] Found \'{n}\' hits'.format(n = hits))
    max_pages = int(hits / 100) + 1
    pages = 10 if max_pages > 10 else max_pages
    logging.info('[core api] Got \'{max_pages}\' pages with search
results, using \'{pages}\' of them'
                 .format(max_pages = max_pages, pages = pages))

    progressSignal.emit(0, pages)

    ids = []
    for page in range(pages):
        logging.info('[core api] Retrieving page:
\'{page}\''.format(page = page + 1))
        statusSignal.emit('Retrieving page: \'{page}\''.format(page =
page + 1))

        try:
            articles =
api_request('articles/search/{query}'.format(query=query),
                                        {'pageSize': 100, 'page': page + 1,
                                        'metadata': 'false', 'fulltext':
'false', 'citations': 'false',
                                        'similar': 'false', 'duplicate':
'false', 'urls': 'false', 'faithfulMetadata': 'false'})

            for rec in articles['data']:
                ids.append(rec['id'])

        except GeneralHTTPException:
            logging.error('[core api] GeneralHTTPException occurred
while retrieving page: \'{page}\''.format(page = page))

        progressSignal.emit(page + 1, pages)
    return ids


def get(id):
    return api_request('articles/get/{id}'.format(id=id), {})['data']


class GeneralException(Exception):
    pass

class NoConfigFile(Exception):
    pass

class NoApiKeyInConfigFile(Exception):
    pass

class GeneralHTTPException(GeneralException):
    pass

class MalformedJSONRequest(GeneralHTTPException):
```

```
    pass

class InvalidAPIKey(GeneralHTTPException):
    pass

class TooManyQueriesInRequest(GeneralHTTPException):
    pass

class TooManyRequests(GeneralHTTPException):
    pass

class ProxyError(GeneralHTTPException):
    pass

class ResponseTimeout(GeneralHTTPException):
    pass

class PageNotFound(GeneralHTTPException):
    pass

class InternalServerError(GeneralHTTPException):
    pass

class GeneralResponseException(GeneralException):
    pass

class MissingParameter(GeneralResponseException):
    pass


import requests
from .exceptions import GeneralHTTPException, MalformedJSONRequest,
InvalidAPIKey, \
                        TooManyQueriesInRequest, TooManyRequests,
ProxyError, ResponseTimeout, \
                        PageNotFound, InternalServerError

import logging
import time

C_QUERY_FORMAT = 'https://core.ac.uk:443/api-v2/{api_cmd}'

g_api_key = ''

def update_api_key(api_key):
    global g_api_key
    g_api_key = api_key

def api_request(api_cmd, params):
    global g_api_key

    params['apiKey'] = g_api_key
    url = C_QUERY_FORMAT.format(api_cmd=api_cmd)
```

```
    logging.info('[core api] Request transmitting:')
    logging.info('[core api] url: \'{url}\' params:
\'{params}\''.format(url = str(url).encode('ascii', 'ignore'),

params = str(params).encode('ascii', 'ignore')))

    for request_try in range(3):
        time_before = time.time()
        response = requests.get(url, params=params, timeout=60)
        logging.info('[core api] Try n: {n} completed in: \'{time:.3}\'
sec'
                     .format(n = request_try + 1, time = time.time() -
time_before))

        logging.info('[core api] Status code: {code}'.format(code =
response.status_code))

        if response.status_code == 200:
            return response.json()

        time.sleep(5)

    if response.status_code == 400:
        logging.error('[core api] Got MalformedJSONRequest status code
in response!')
        raise MalformedJSONRequest
    elif response.status_code == 401:
        logging.error('[core api] Got InvalidAPIKey status code in
response!')
        raise InvalidAPIKey
    elif response.status_code == 403:
        logging.error('[core api] Got TooManyQueriesInRequest status
code in response!')
        raise TooManyQueriesInRequest
    elif response.status_code == 429:
        logging.error('[core api] Got TooManyRequests status code in
response!')
        raise TooManyRequests
    elif response.status_code == 502:
        logging.error('[core api] Got ProxyError status code in
response!')
        raise ProxyError
    elif response.status_code == 408:
        logging.error('[core api] Got ResponseTimeout status code in
response!')
        raise ResponseTimeout
    elif response.status_code == 404:
        logging.error('[core api] Got PageNotFound status code in
response!')
        raise PageNotFound
    elif response.status_code == 500:
        logging.error('[core api] Got InternalServerError status code
```

```
in response!')
        raise InternalServerError

    logging.error('[core api] Got unknown status code in response!')
    raise GeneralHTTPException
```

```
import core_api.articles
import logging
import pandas as pd
from statistics import mean


def authors_hits(authors_series, progressSignal, statusSignal):
    # explode series of 1d arrays to list
    # and filter out empty authors
    authors = [author for authors_arr in authors_series for author in
authors_arr if author]

    logging.info('Authors n: \'{n}\''.format(n=len(authors)))
    # remove duplicates
    authors = list(dict.fromkeys(authors))
    logging.info('Unique authors found:
\'{n}\''.format(n=len(authors)))

    progressSignal.emit(0, len(authors))

    # find hits in core for every author
    core_hits = []
    for index, author in enumerate(authors):
        statusSignal.emit('Retrieving rating for author
\'{author}\''.format(author = author))
        core_hits.append(core_api.articles.get_author_hits_n(author))
        progressSignal.emit(index, len(authors))

    self_hits = []
    for author in authors:
        hit = 0
        for authors_arr in authors_series:
            for raw_author in authors_arr:
                if author == raw_author: hit += 1

        self_hits.append(hit)

    return pd.DataFrame({'self': self_hits, 'core': core_hits}, index =
authors)


def rate(articles, progressSignal, statusSignal):
    authors_rates = authors_hits(articles['authors'], progressSignal,
statusSignal)[['core', 'self']]\
                    .apply(lambda row: row['core'] * row['self'],
axis=1)

    authors_rates = authors_rates.apply(lambda r: r /
authors_rates.max())

    rates = pd.DataFrame(index = articles.index)
```

```
    rates['authors'] = articles['authors'].map(lambda arr:
                                                0 if not arr
                                                else mean(map(lambda
a: authors_rates[a], arr)))

    rates['year'] = articles['year'].map(lambda y: 0 if pd.isna(y) else
y / articles['year'].max())

    rates['contributors'] = articles['contributors'].map(lambda arr:
len(arr)) / \
                                (max(articles['contributors'].map(lambda
arr: len(arr))) + 1)

    reduced_rates = rates.apply(lambda r: r['year']*2 +
r['contributors'] + r['authors']*3, axis = 1)

    return reduced_rates.apply(lambda r: r /
reduced_rates.max()).sort_values(ascending=False)


import logging
import core_api.articles
import json
import pandas as pd
import re
import os.path
from core_api.exceptions import GeneralHTTPException

json_ids_dump_path = 'dumps/{query}_found_articles_ids.json'
json_data_dump_path = 'dumps/{query}_fetched_articles_data.json'
pkl_dump_path = 'dumps/{query}_fetched_articles_data.pkl'


def find_and_fetch_articles(query, progressSignal, statusSignal):
    logging.info('[algo] Searching for query:
\'{query}\''.format(query=query))

    if os.path.isfile(pkl_dump_path.format(query=query)):
        logging.info('[algo] Found cached results for this query, using
them')
        statusSignal.emit('Found cached results for this query')
        progressSignal.emit(1, 1)

        return pd.read_pickle(pkl_dump_path.format(query=query))

    ids = core_api.articles.search(query, progressSignal, statusSignal)
    logging.info('[algo] Found \'{n}\' matching
articles'.format(n=len(ids)))

    with open(json_ids_dump_path.format(query=query), 'w') as
dump_file:
        dump_file.write(json.dumps(ids))
```

```
    progressSignal.emit(0, len(ids))

    articles = []
    for index, id in enumerate(ids[:20]):
        logging.info('[algo] Fetching article id:
\'{id}\''.format(id=id))
        statusSignal.emit('Fetching article id:
\'{id}\''.format(id=id))

        try:
            data = core_api.articles.get(id)
        except GeneralHTTPException:
            logging.error('[algo] Cannot fetch article id:
\'{id}\''.format(id=id))
            continue

        # every author has coma separated firstname and lastname.
        # this is bad for pandas parser
        data['authors'] = [author.replace(',', '') for author in
data['authors']]
        articles.append(data)

        progressSignal.emit(index, len(ids))

    with open(json_data_dump_path.format(query=query), 'w') as
dump_file:
        dump_file.write(json.dumps(articles))

    articles_df = pd.DataFrame(articles)
    articles_df.set_index('id', inplace=True)

    articles_df.to_pickle(pkl_dump_path.format(query=query))

    return articles_df


def filter_gargbage(articles):
    replace_breaks = lambda t: t.replace('\n', ' ').replace('\r', '
').replace('\t', ' ')

    # remove raw unicode codes from text
    articles['authors'] = articles['authors'].apply(lambda arr:
[re.sub(r'\\u\d.', ' ', a) for a in arr])
    # remove trailing spaces
    articles['authors'] = articles['authors'].apply(lambda arr:
[a.strip() for a in arr])
    # replace \n \t \r with spaces
    articles['authors'] = articles['authors'].apply(lambda arr:
[replace_breaks(a) for a in arr])
    # replace whitespace duplicates with single space
    articles['authors'] = articles['authors'].apply(lambda arr: ['
'.join(a.split()) for a in arr])
    # some authors have 'null' values, dismiss them
```

```
    articles['authors'] = articles['authors'].apply(lambda arr: [a for
a in arr if 'null' not in a.split()])
    # filter empty authors cells
    articles['authors'] = articles['authors'].apply(lambda arr: [a for
a in arr if a])
    # need to interpret list as a set cause later pandas may hash that
column
    articles['authors'] = articles['authors'].apply(frozenset)

    articles['title'] = articles['title'].apply(lambda t: t if 'null'
not in t.split() else pd.NA)
    articles['title'] = articles['title'].apply(replace_breaks)
    articles['title'] = articles['title'].apply(lambda t: '
'.join(t.split()))

    articles = articles[articles['authors'].map(len) > 0]
    articles = articles[articles['title'].map(pd.isna) == False]

    articles = articles.drop_duplicates(['authors', 'title'])
    return articles
```

```
__all__ = ['utils']


from algo import routines, rating
from PyQt5.QtWidgets import QTableWidgetItem
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import *
import webbrowser
import logging
from threading import Lock

redrawLock = Lock()
threadpool = QThreadPool()


class SignalsWrapper(QObject):
    progress_signal = pyqtSignal(int, int)
    status_signal = pyqtSignal(str)
    results_signal = pyqtSignal(object, object)


class SearchRunnable(QRunnable):

    def __init__(self, query, signals_wrapper, *args, **kwargs):
        super(SearchRunnable, self).__init__()
        self.query = query
        self.signals_wrapper = signals_wrapper

    @pyqtSlot()
    def run(self):
        articles = routines.find_and_fetch_articles(self.query,
self.signals_wrapper.progress_signal,

self.signals_wrapper.status_signal)
        articles = routines.filter_gargbage(articles)
        ratings = rating.rate(articles,
self.signals_wrapper.progress_signal,
self.signals_wrapper.status_signal)
        self.signals_wrapper.results_signal.emit(articles, ratings)


def tbl_cell_clicked(item):
    if item.column() == 3:
        webbrowser.open(item.text())


def progress_bar_handler(w, val, max):
    with redrawLock:
        w.prgbrStatus.setValue(val)
        w.prgbrStatus.setMaximum(max)
```

```
def status_label_handler(w, status):
    with redrawLock:
        w.lblStatus.setText(status)


def results_handler(w, articles, ratings):
    with redrawLock:
        w.btSearch.setEnabled(True)
        w.prgbrStatus.setValue(0)
        w.lblStatus.setText('Done')

        w.tblResults.clear()
        w.tblResults.setRowCount(len(ratings.index))
        w.tblResults.setColumnCount(4)
        w.tblResults.setHorizontalHeaderLabels(['Rate', 'Title',
'Authors', 'URL'])
        w.tblResults.setCurrentCell(0, 0)

        download_icon = QIcon('gui/res/download.png')

        logging.info('Rates: ')
        for id, rate in ratings.items():
            row_index = w.tblResults.currentRow()

            w.tblResults.setItem(row_index, 0,
QTableWidgetItem('{rate}%'.format(rate=int(rate * 100))))
            w.tblResults.setItem(row_index, 1,
QTableWidgetItem(articles.at[id, 'title']))
            w.tblResults.setItem(row_index, 2, QTableWidgetItem(',
'.join(articles.at[id, 'authors'])))
            w.tblResults.setItem(row_index, 3,
QTableWidgetItem(download_icon, articles.at[id, 'downloadUrl']))

            w.tblResults.setCurrentCell(row_index + 1, 0)

            logging.info('rate: \'{rate:.3}\' id: \'{id}\' authors:
\'{authors}\' title: \'{title}\' url: \'{url}\''
                         .format(rate=rate, id=id,
authors=list(articles.at[id, 'authors']),
                                 title=articles.at[id, 'title'],
url=articles.at[id, 'downloadUrl']))

def bt_search_handler(w):
    w.btSearch.setEnabled(False)
    w.tblResults.itemClicked.connect(tbl_cell_clicked)

    signals_wrapper = SignalsWrapper()
    signals_wrapper.progress_signal.connect(lambda val, max:
progress_bar_handler(w, val, max))
    signals_wrapper.status_signal.connect(lambda status:
status_label_handler(w, status))
    signals_wrapper.results_signal.connect(lambda articles, ratings:
```

```
results_handler(w, articles, ratings))

    threadpool.start(SearchRunnable(w.txtSearch.text(),
signals_wrapper))


from PyQt5 import uic
from . import handlers
from PyQt5.QtCore import *


def load_ui():
    return uic.loadUi('gui/res/mainwindow.ui')


def register_handlers(w):
    w.btSearch.clicked.connect(lambda : handlers.bt_search_handler(w))
```

```
import logging
from datetime import datetime
from gui import utils
from PyQt5 import QtWidgets
import sys


def main():
    log_path = 'logs/portoporto_{time}.log'.format(time =
datetime.now().strftime("%d_%m_%Y__%H_%M_%S"))
    logging.basicConfig(filename=log_path, level=logging.INFO)

    app = QtWidgets.QApplication(sys.argv)
    w = utils.load_ui()
    utils.register_handlers(w)
    w.show()
    app.exec()


if __name__ == '__main__':
    main()
```

ДОДАТОК 3

ПЛАТФОРМА ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В СФЕРІ МІЖНАРОДНОГО НАУКОВО-ТЕХНІЧНОГО СПІВРОБІТНИЦТВА

Опис програми

УКР.НТУУ“КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ ТР-62137_20Б

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Платформа для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва дозволяє користувачу програмного забезпечення здійснювати вузьконаправлені запити на пошук потрібної інформації науково-технічної галузі та отримувати відповідь у вигляді послідовності, впорядкованих за релевантністю, публікацій з позначеними авторами та режимом доступу до кожної з них.

Інтерфейс користувача дозволяє інтуїтивно виконувати користування програмним додатком. Запит на пошук вводиться в пошуковий рядок, та, одразу після обробки запиту, результат виводиться у вікно під пошуковим рядком. Інформація розподілена за власними типами по колонках: відсоток релевантності, назва публікації, автори, посилання на публікацію.

Пошуковий рядок дозволяє використовувати синтаксис еластичного пошуку з метою здійснення вузьконаправлених запитів.

В розробці було використано: програмну платформу мови розробки Qt Framework, мову розробки Python, прикладний програмний інтерфейс CORE API, засіб локального збереження інформації pandas, програмну платформу побудови інтерфейсу користувача PythonQt.

ЗМІСТ

-4-
ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до назви дипломної роботи, програма реалізує платформу, що підтримує прийняття рішень, шляхом надання можливості пошуку інформації з науково-технічної галузі та отримання її у, впорядкованій за релевантністю, послідовності.

Платформа для пошуку та вилучення інформації, та додаток обробки інформації працюють як єдиний програмний додаток.

-5-
ПРИЗНАЧЕННЯ

Реалізація платформи для забезпечення підтримки прийняття рішень в сфері міжнародного науково-технічного співробітництва покликана прискорити прийняття рішення в межах аналітичних досліджень міжнародного співробітництва. Аналітики отримують релевантну інформацію за вузьконаправленими запитами, тому не втрачають час на її впорядження та відокремлення більш релевантної від менш релевантної.

Платформа вміє працювати з синтаксисом еластичного пошуку з метою забезпечення можливості здійснення вузьконаравленого запиту.

Релевантність знайденої інформації встановлюється в результаті обробки за критеріями:

- Кількість робіт автора за запитом
- Кількість згадувань автора у БД
- Кількість осіб, задіяних у публікації (контриб’юторів)
- Давність публікації

-6-
ОПИС СТРУКТУРИ

Програмний продукт складається з трьох частин. Це платформа для пошуку, вилучення та збереження інформації, та додаток, що впорядковує інформацію за релевантністю та інтерфейс користувача.

Користувач виконує введення запиту у пошуковий рядок. Після натискання кнопки пошуку запит відправляється платформою через прикладний програмний інтерфейс CORE API до бази даних CORE. Після обробки запиту до платформи повертається відповідь з даними про знайдені за запитом публікаці. Дані зберігаються у фреймах засобу збереження інформації pandas. Далі додаток обробки інформації виконує оцінку даних кожної публікації та формує рейтингову оцінку, за якою публікації будуть впорядковані у релевантну послідовність та виведені у вікно відображення знайденої інформації у інтерфейсі користувача.

У інтерфейсі користувача присутні елементі:

1. Рядок введення запиту
2. Вікно відображення знайденої інформації
3. Таблиця з колонками, що позначають поле відображаємої інформації (Відсоток релевантності, Тема, Автор, Посилання)
4. Елемент відображення поточного стану роботи програмного додатку

-7-
ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для запуску програмного додатку необхідно, щоб на машину було встановлено: Python версії не нижче 3.4; PyQt5; PyCharm; pandas.

Для запуску інтерфейса користувача необхідно, щоб на машину було встановлено QtCreator.

-8-
ВХІДНІ І ВИХІДНІ ДАНІ

Вхідні дані:

- запит, введений користувачем

Вихідні дані:

- результат пошуку, здійсненого за запитом